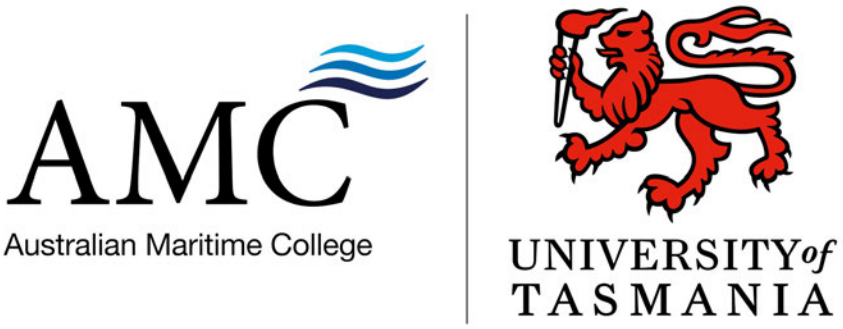

GAUSSIAN PROCESSES APPLIED TO SYSTEM IDENTIFICATION, NAVIGATION AND CONTROL OF UNDERWATER VEHICLES

Wilmer Ariza Ramirez

Australian Maritime College

Submitted in fulfilment of the requirements
for the Degree of Doctor of Philosophy

Australian Maritime College
University of Tasmania
August, 2019

This page is intentionally left blank.

Declaration of Originality

I declare that this thesis contains no material which has been accepted for a degree or diploma by the University or any other institution, except by way of background information and duly acknowledged in the thesis, and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due acknowledgement is made in the text of the thesis.

Signature

Wilmer Ariza Ramirez

Date:July 9, 2019

This page is intentionally left blank.

Authority access

Parts of this thesis are not to be made available for loan or copying for two years following the date this statement was signed. Following that time the thesis may be made available for loan and limited copying and communication in accordance with the *Copyright Act 1968*

Signature

Wilmer Ariza Ramirez

Date:July 9, 2019

This page is intentionally left blank.

Statement regarding published work

The publishers of the papers comprising Chapters 2 to/and 5 and Chapter 7 hold the Copyright for that content and access to the material should be sought from the respective journals. The remaining non published content of the thesis may be made available for loan and limited copying and communication in accordance with the *Copyright Act 1968*

Signature

Wilmer Ariza Ramirez

Date:July 9, 2019

This page is intentionally left blank.

Statement of Co-Authorship

The following people and institutions contributed to the publication of work undertaken as part of this thesis:

- Wilmer Ariza Ramirez, Australian Maritime College/University of Tasmania, **Candidate**.
- Dr. Zhi Quan Leong, Australian Maritime College/University of Tasmania, **Author 1**.
- Dr. Hung Nguyen, Australian Maritime College/University of Tasmania, **Author 2**.
- Prof. Dr. Juš Kocijan,Department of Systems and Control/ Jožef Stefan Institute and School of Engineering and Management/ University of Nova Gorica, **Author 3**.
- Dr. Shantha Gamini Jayasinghe, Australian Maritime College/University of Tasmania, **Author 4**.

Paper 1, Non-parametric Dynamic System Identification of Ships Using Multi-Output Gaussian Processes.

Located in Chapter 2, Candidate was the primary author and contributed 75% to the planning and preparation of the research project and subsequent paper. Authors 1,2,3 and 5 contributed to the analysis and interpretation of the research data and contributed to the interpretation of the work by critically revising the paper

Paper 2, Non-parametric Dynamic System Identification of Underwater Vehicles Using Multi-Output Gaussian Processes

Located in Chapter 3, Candidate was the primary author and with author 3 contributed to the planning and preparation of the research project and subsequent paper. Candidate contributed approximately 60% to the planning, execution and preparation of the work for the paper. Authors 1,2,3 and 4 contributed to the analysis and interpretation of the research data and contributed to the interpretation of the work by critically revising the paper.

Paper 3, Machine Learning Post Processing of Underwater Vehicle Pressure

Sensor Array for Speed Measurement.

Located in Chapter 4, Candidate was the primary author and contributed 75% to the planning and preparation of the research project and subsequent paper. Authors 1,2, and 4 contributed to the analysis and interpretation of the research data and contributed to the interpretation of the work by critically revising the paper.

Paper 4, Navigation of underwater vehicles with Gaussian processes unscented Kalman filter.

Located in Chapter 5, Candidate was the primary author and contributed 75% to the planning and preparation of the research project and subsequent paper. Authors 1,2, and 4 contributed to the analysis and interpretation of the research data and contributed to the interpretation of the work by critically revising the paper.

Paper 5, Exploration of the Applicability of Probabilistic Inference for Learning Control on Underactuated Autonomous Underwater Vehicles.

Located in Chapter 7, Candidate was the primary author and contributed 75% to the planning and preparation of the research project and subsequent paper. Authors 1,2, and 4 contributed to the analysis and interpretation of the research data and contributed to the interpretation of the work by critically revising the paper.

Signed: _____

Dr. Zhi Quan Leong
Primary Supervisor
Australian Maritime College
University of Tasmania
Date:July 9, 2019

Signed: _____

Dr. Hung Nguyen

Co-Supervisor
Australian Maritime College
University of Tasmania
Date:July 9, 2019

Signed: _____

Dr. Shantha Gamini Jayasinghe

Co-Supervisor
Australian Maritime College
University of Tasmania
Date:July 9, 2019

Signed: _____

Dr. Vikram Garaniya

Acting Director NCMEH
Australian Maritime College
University of Tasmania
Date:July 9, 2019

This page is intentionally left blank.

ABSTRACT

Autonomous underwater vehicles (AUVs) are increasingly being used in commercial, military and scientific organisations for various applications ranging from hull inspections to underwater explorations. Despite the steady growth in applications, communication with the AUVs has been the major challenge, mainly due to the cost and limited availability of underwater communication system. Therefore, AUVs are required to have a higher degree of autonomy compared to the land based and aerial autonomous vehicles. In addition to the lack of communication, AUVs cannot measure their position or orientation with high accuracy since many states are calculated or corrected based on internal observers which employ other measurements and mathematical models. Popular methods of motion control and inertial navigation for AUVs require costly experiments as the mathematical representation of the vehicle is used for the prediction and correction of states, controller deduction and the tuning procedure.

New non-parametric algorithms for inertial navigation and motion control are required as the standard techniques employ a very complex mathematical model of the vehicle that usually is just an approximation as it is calculated based on ideal conditions. New non-parametric models can lead to the improvement of position estimation and robust control of a vehicle as these models are estimated over real data from the platform and they can include models of perturbances. Machine learning has become a principal tool in robotic applications as drones, autonomous vehicles and earth bioinspired robots. A specific method of machine learning that can be employed for the construction of non-parametric models is Gaussian Processes (GPs). GPs are based on the theory that natural phenomena can be fitted to a multivariable Gaussian distribution.

This research started with an investigation to check whether Multi-output Gaussian Processes are capable of producing a non-parametric GPs model for ships and underwater vehicles with the advantage of being able to represent the relation between the outputs. In a similar way, it was researched whether GPs can be employed for post-processing of pressure sensors data to estimate the vehicle speed without other types of sensors. Another application explored was the application of GP-UKF (Gaussian processes unscented Kalman Filter) for navigation. GP-UKF was successfully simulated to navigate

an underwater vehicle without the need to recognize a mathematical model as the model used is a GPs model identified in previous experiments. Finally, GPs were employed to learn a policy for motion control of an underwater vehicle via the application of model-based reinforced learning based on the use of a GP model. Reinforced learning with GPs model was shown to be a faster option than non-model standard reinforced learning techniques with less quantity of experiments and calibration required over the platform to learn a policy to control underwater vehicles.

The results presented in this thesis prove that GPs are able to model autonomous underwater vehicle in an effective way by the application of multi-output GPs, allowing a series of possible applications in underwater vehicles. From all the possible applications, this thesis explored the application for post-processing of sensors data for measurement of speed-based on the difference of pressure around the head of the vehicle while the vehicle is in motion. Also, it was researched their employment as the internal model for inertial navigation algorithms such as the unscented Kalman filter for improvement in the prediction and correction of measurement. A final application of GPs for model based reinforced learning algorithms was as a reference model to allow the learning of policies for motion control for underwater vehicles.

The contributions of this thesis start by establishing that GPs are capable of effectively creating a non-parametric model-based on Gaussian distribution from an underwater vehicle data. Additionally, it is established that GPs non-parametric models are a solution for the creation of transfer functions between raw sensor data as a pressure sensor array to vehicle speed for AUVs. It has been shown that GPs non-parametric models from underwater vehicles allow more robust navigation system by their integration as the internal model in an unscented Kalman filter. Also, their application to navigation to allow a faster calibration by removing the requirement of tuning of the variance matrices of state and measurement as these matrices are produced by the non-parametric model. A final contribution of this thesis is the use of GPs non-parametric models for model based reinforced learning by the search of policies for waypoint tracking and path tracking with a low quantity of trials require and overperforming common robust PID controllers.

ACKNOWLEDGEMENTS

I would like to acknowledge the help from my supervisors Dr Zhi Leong, Dr Hung Nguyen and Dr Shantha Gamini Jayasinghe, who made this work possible. Their friendly guidance and expert advice have been invaluable throughout all stages of the work. I would also wish to express my gratitude to Professor Juš Kocijan for all valuable suggestions which have contributed greatly to the improvement of the thesis.

It is a pleasure to thank my friends Cristina, Kevin, Paola, Cesar, Iris, Karen and Eduardo, for the wonderful times we shared ,giving me the necessary distractions from my research and listening ear.

The people with the greatest indirect contribution to this work are my mother, Carmen Rosa Ramirez, who taught me a love for mathematics and my father, Gabriel Ariza, who taught me a love for engineering. I want to thank them, as well as my brother Leonardo, my wife Kelly, and my little son Ian for their constant encouragement.

Professional editor, Bron Fionnachd-Fein, provided copyediting and proofreading services, according to the guidelines laid out in the university-endorsed national 'Guidelines for editing research theses'

This page is intentionally left blank.

TABLE OF CONTENTS

This page is intentionally left blank.

LIST OF TABLES

LIST OF FIGURES

This page is intentionally left blank.

Chapter 1

Introduction

"If a machine is expected to be infallible, it cannot also be intelligent."

Alan Turing

1.1 Background

An autonomous underwater vehicle (AUV) is a specialised robotic platform with either a relative or absolute degree of autonomy [1]. The number of AUVs in operation is ever increasing due to the growing needs of commercial, military and scientific organisations [2, 3]. This need is dictated by the inherent nature of AUVs to meet especially challenging mission requirements in high-risk (and usually unknown) environments with the absence of global positioning system (GPS) signal, high pressure due to water depth, limited energy, and long-time missions without direct communication to the vehicle. AUVs can be categorised in relation to the degree of motion control obtainable over each degree -of -freedom of the vehicle. An AUV with multiple thrusters and fins can be fully actuated or underactuated, and these AUVs reach the category of fully actuated if the control system of thrusters and fins can control and move the vehicle in any direction without direct coupling to other degrees of freedom. On the other hand, underactuated AUVs are vehicles with a limited number of thrusters and fins. Underactuated vehicles cannot control all their movement directions as some movements are directly coupled to opposite control actuators. The most common configuration for this type of AUV is a single thruster with two pairs of fins on a torpedo-shape form, which is configured for control over three degrees of freedom (surge, yaw, pitch) [3]. This configuration of underactuated AUV is the most popular as they are the most energy-efficient design, in spite of the highly complex control required.

A critical step in achieving a degree of autonomy is the correct implementation of navigation algorithms and motion-control algorithms capable of locating the vehicle in space and directing the vehicle to a waypoint or path designed by an internal or external mission planner. In general, the process of designing a motion controller can be divided into three steps: identification, motion-controller design, and tuning. In the case of AUVs, the controller needs to be paired with an inertial navigation system. Both the controller and navigation system commonly require knowledge of hydrodynamic coefficients to create a mathematical model. he precision and accuracy of the navigation system will be reflected in the results of the motion-control algorithm.

The inertial navigation system of underwater vehicles is of high importance. In the

core of the navigation system, an inertial measurement unit composed of gyroscopes, accelerometers and compasses is employed to calculate the vehicle orientation and velocities [4]. However, this configuration has a weakness, i.e. the integration of the kinematic acceleration signal will produce a drift of the resultant position. This can be overcome by the use of other sensors such as a Doppler velocity log (DVL). However, for many missions it is not possible to measure the vehicle position or velocities as there is not a surface against which the measurement can be made with DVL sensors. The prediction and correction of a variable based on the measurement from an inertial motion unit are conducted usually by observers who employ an internal mathematical model to predict the vehicle state and overcome noise. Kalman filters such as the non-linear extended Kalman filter and the unscented Kalman filter are commonly employed [5].

The motion control of underwater vehicles can be categorised by the application of specific methodologies. In commercial vehicles, linear controllers are commonly used such as PID and linear quadratic regulator [6]. Other techniques such as linear quadratic Gaussian that overcome signals with Gaussian noise or incomplete state information have been explored [7]. Non-linear methodologies such as sliding mode controller [8] and methodologies that use Lyapunov and backstepping techniques have been proposed as possible robust solutions [9]. All these methodologies of motion control, linear or non-linear, require the use of a mathematical model for the formulation of the controller or at least for calibration as this step can also take considerable time. Furthermore, field experiments for calibration can put the vehicle at additional risk.

The quality of the internal mathematical model commonly is directly linked to the precision of the inertial navigation and this will reflect on the accuracy of the controller and success of a mission. Any reduction of noise over the measurement will lessen the grade of robustness required in the motion controller. In the same way, a good prediction of the position will reduce the need for ascending to the surface to obtain GPS signal to correct the drift of the inertial navigation system [10]. Similarly, non-linear controllers employ coefficient-based models as part of the controller equations. If the calculated values are incorrect, the error can affect the performance of the vehicle and can produce undesired results, possibly leading to loss of vehicle control [2].

A solution to this problem of the calculation of coefficients for a formulated mathematical model is the use of machine learning algorithms. Machine learning algorithms have been employed to identify hydrodynamic coefficients or to compensate for unknown variables. The preferred machine learning family of algorithms has been neural networks. Different neural network controller schemes have been suggested and implemented in the past [11], some of which have been particularly applied to the underwater vehicle control problem, e.g. supervised control; direct inverse control; model reference control; critic control; internal model control; and predictive control.

Robotics research for autonomous vehicles has focused in recent years on the application of machine learning algorithms for a greater grade of autonomy by learning from experience. This research has developed novel techniques that do not require previous knowledge of the platform for navigation or control. However, robotic research has primarily focused on machine learning to reduce the requirement of mathematical models for land and air vehicles while research for AUVs has lagged behind as many algorithms have difficulties in being applied to underwater platforms. In many cases, the research for underwater platforms has been in the application of neural network methodologies. However, there are other types of algorithms that can be employed. A Gaussian Process (GP) is a technique of machine learning that belongs to the class of black-box models. GPs are based on the use of probabilistic functions to fit a set of data. GPs differ from most other approaches in that they do not try to approximate the modelled system by fitting the parameters of the selected basis functions, but rather they search for the relationship among the measured data [12].

GPs have a series of benefits that can be useful for application in underwater vehicles:

1. smaller number of identification data is required;
2. prior knowledge about the vehicle can be included;
3. work with data that are noisy; and
4. the prediction model contains a measure of confidence.

In the case of underwater vehicles, the ability of GPs to overcome noisy platforms and the measurements of confidence are advantages that can be exploited in navigation and control. In navigation, the confidence measurement can be employed to produce the

covariance matrices of the Kalman filter to correct and predict the measurement. GPs can also be employed in model-based RL (reinforced learning). Model-based RL employs an internal model to find a policy capable of controlling a vehicle by using a gradient descent methodology. GPs can generate a mathematical model of an AUV with a small quantity of episodes, resulting in faster learning.

1.2 Research Questions

The aims of the project are to investigate the applicability of GPs to system identification of underwater vehicles for the production of a non-parametric model based in data that can be measured and explore some application that potentially can benefit from the use of non-parametric models. Thus, the specific research question for this project is:

Can GPs be used for system identification of AUVs and what applications can benefit from the use of a non-parametric model?

As dynamic identification with GPs has never been applied to marine vehicles, an initial set of simulation with a container ship was employed to develop a methodology. The lesson learned allowed the implementation for underactuated AUVs. After the successful application of GPs for system identification, the possible applications selected were based in the development of the platform MULLAYA (MULLAYA AUV is a platform for engineering research of underwater vehicles from the Australian Department of Defence Science and Technology) and the experiments or simulations that the platform at the moment is capable of execute.

1.3 Scope and Limitations of the Project

The scope of this project refer to the innovative application of GPs to obtain a dynamic non-parametric model of underwater vehicles and the application of such models to improve navigation and control of underwater vehicles. The limitation of this project are based on the limitations of possible experiments, simulation scenarios, future development and safety requirements of the platform MULLAYA.

1.4 Novel Aspects of the Research

The novel aspects of this research are derived from the use of GPs for the application of GPs into underwater vehicle identification, navigation and control. The research has explored several possibilities for the application of GPs, and the results can be categorised as:

1. This is a pioneering study where for first time it is applied the dynamic system identification with multi-output GPs to underactuated ships;
2. This study is the first instance where a non-parametric model who maintain the relation between the outputs of system is obtained by the application of multi-output GPs to underactuated AUVs;
3. This study for the first time shows the application of machine learning methodologies for the post-processing of pressure sensors data for surge speed measurement of underwater vehicles;
4. Employment for the first time of GP-unscented Kalman filter for prediction and correction of position and orientation of underwater vehicles; and
5. For the first time on this study is shown that reinforced learning can be applied to underactuated AUVs with a small quantity of experiments with the application of non-parametric model based reinforced learning control.

These applications of GPs were chosen in accordance of the present and future development of the platform MULLAYA.

1.5 Outline of Thesis

This thesis consists of scientific papers supplemented by supporting chapters, which follow a structure as outlined below.

Chapter 2: Introduces the multi-output GPs as a solution for system identification(SI) of marine vehicles. A simulation of a container ship model is employed to obtain data to train the GPs model, and a series of quality parameters are employed to measure the reliability.

Chapter 3: Introduces the multi-output GPs as a solution for SI of underwater vehicles. A simulation of a REMUS 100 model is employed to obtain data to train the GPs model, and a series of quality parameters are employed to measure the reliability.

Chapter 4: Shows the applicability of machine learning algorithms to post processes pressure measurement from an underwater vehicle to predict the surge velocity.

Chapter 5: Outlines and discusses the application of GP-UKF to predict and correct the measurements from the inertial navigation system of an underwater vehicle.

Chapter 6: Shows the semi-empirical calculation of the hydrodynamic coefficients of MULLAYA AUV and its comparison with experiments.

Chapter 7: Explore the applicability of model-based reinforced learning with a GPs model to learn control policies for waypoint tracking, depth control and path tracking.

Chapter 8: Presents conclusions drawn from the study and suggests avenues for future research.

This page is intentionally left blank.

Chapter 2

Non-parametric Dynamic System Identification of Ships Using Multi-Output Gaussian Processes

This chapter present a first apporach to system identification of underactuated AUVs by the application to a more simple underactuated ship model and it has been published in the *Journal of Ocean Engineering* [13].

2.1 Introduction

Dynamic modelling of oceanic vehicles including surface ships, semisubmersibles/ submersible platforms, and unmanned underwater vehicles is an active research field due to the application importance of these vessels such as goods transport, oil and gas exploration [14], underwater survey, and fishery. The common approach to modelling such vehicles is the use of Newtonian-Lagrangian mathematical models which are usually predefined. However, the presence of unaccounted dynamics caused by parametric and non-parametric uncertainties in a predefined model can increase the error between the predicted output and the real output. The cause of these uncertainties is commonly attributed to ocean currents, waves, wind, and hydrodynamic interaction with nearby structures. Since oceanic vehicles operate in dynamically changing environments performance of traditional controllers such as PID, LQR, and backstepping controllers [15, 16] degrade over time of operation as they require an initial offline design, calibration and are directly dependent on the predefined system parameters. An optional approach to predefined mathematical modelling is the use of non-parametric system identification (SI) methods. In this context, the application of modern machine learning algorithms that are capable of producing evolutionary adaptability to the environment has been identified as a promising approach for SI [17]. The present study focuses on its application for the identification of surface ship dynamics.

There are multiple mathematical models for the representation of ships dynamics. Some models are 3 DoF models where the surge, sway and yaw are represented by linear and nonlinear equations [18]. Other more advanced models such as [19]used a 4 DoF nonlinear model for ships including the rolling effect. The dynamic modelling of ship is a prerequisite for the design of its autopilot, navigation, steering control, and damage identification systems. The exactitude of the model can lead to the reduction of fuel consumption [20] by the correct tuning of an autopilot, better vehicle stability, and less stress over the vehicle structure [21] and the possibility of advanced algorithms such as automatic ship berthing [22].

Dynamic mathematical models are usually obtained by the application of Newtonian and Lagrangian mechanics, which lead to a complex system of coupled equations defined

by a series of parameters. The parameters are the representation of added masses, hydrodynamics damping constants, and constants related directly with control forces such as propellers and rudders. Over the years, multiple methods have been developed to determine the hydrodynamic parameters of ships, e.g. empirical formulas, captive model test, computational fluid dynamics (CFD) calculation and parameter estimation based in SI. The most recognized and accepted method is captive model test with planar motion mechanics [23]. This method requires the use of sophisticate facilities such as towing tanks, rotating arms and planar motion mechanism to produce the required ship manoeuvres that allow the parameters to be identified. These manoeuvres can also be replicated virtually via CFD which can be a more affordable option [24]. However, as the accuracy of CFD is highly dependent on the numerical settings and requires validation, physical experiments are still preferred over computational solutions.

Parameter estimation based in SI methodologies offer a practical way to identify the hydrodynamic parameters of a ship model or a complete model. The data source for SI can be free-running model tests or full-scales trials of existing ships. SI can be categorized in two groups, parametric and non-parametric identification. Parametric identification is based on the use of numerical methods to obtain the hydrodynamics parameters of proposed mathematical models with unknown parameters. Alternatively, non-parametric identification is based on the use of single or multiple kernel functions to create a non-physics related mathematical model which is tuned by a learning procedure that uses data obtained from the original system. Methods like Extended Kalman Filter [25, 26], Unscented Kalman Filter[27], Estimation-Before-Modelling [28], and Backstepping [29] are the most popular numerical methods for coefficient estimation. However, these methods can suffer from linearization and convergence errors. Therefore, more advanced SI methods from machine learning, e.g. neural networks [30], and support vector machines [31] had found their space in parametric ship SI with the use of specific structures (NN) or specific selection of kernel functions(SVM), these specific structure allow the techniques to calculate some coefficients. The principal disadvantage of parametric system identification is the need of controlled test with low external perturbations and specific procedures to reduce the interference and nonlinearities between degrees of freedom.

In contrast to the parametric SI, non-parametric SI has the capacity to learn a complete

model without prior knowledge of the system structure. This learning procedure leads to a simpler model with fewer parameters. Non-parametric SI brings the possibility of incorporating online learning giving the ability to improve the adaptability of the model. The capability to adapt to change is very important for application of evolutionary control techniques and damage identification. The most recognized method of non-parametric identification for ships is recursive neural network (RNN). RNN differs over standard neural networks in the aspect that the structure of the network is organized hierarchically applying the same set of weights recursively over the structure, to produce a scalar prediction on it. [32]. This method has been used with success to identify complex ship designs like catamarans with the final purpose of offline simulation of ship behaviours [33]. [34] presented a modified version of SVMs to capture the full coupled system in four degrees of a ship following a similar methodology to RNNs. The difference between the SVM and the neural network methods is that the SMV is less prone to overfitting, thus can reach a global optimum and require less memory. Wang's proposed a white, grey and black box system, the black box is the result of the mathematical analysis of the grey black box that leads them to recognize an applicable kernel. The drawback of neural networks and SVM machine learning methods is the lack of confident measures, and thus, an error in the prediction cannot be corrected.

Depending on the budget and availability of infrastructure and time, the parametric or non-parametric model characterization can be chosen for a given system. In the case of new designs with low complexity, the parametric identification can be carried out without inconvenience as scale model can be produced and computational CAD files are available. However, for old oceanic vehicles that require fitting of new technology, vehicles that require operation in evolving environments, and vehicles with complex designs the use of non-parametric methods can be more practical.

Nevertheless, not all possible methods of machine learning had found their way to dynamic SI of ships. If a neural network is used to generate a non-parametric model with the inclusion of the variance, the number of hidden units ideally has to be taken to infinity, in which case it turns that a neural network with infinite hidden layers is equivalent to another machine learning method known as Gaussian Processes [35]. GPs is a well-established method in fields such as geostatistics, where the GPs method is

renamed 'kriging' [36]. In GPs based SI the model is built over input-output data and a covariance function is used to characterise the ship behaviour. The advantage of GPs is their ability to work with small quantities of data and noisy data, and the predicted results consist of a mean and variance value. The variance of a future prediction can be used for other purposes as well such as control and model based fault detection since it contains a measure of confidence. [37] and [38] described the application of GPs for the identification of nonlinear dynamics system and provided examples over simple input and single outputs systems. The standard technique of modelling multi-output systems as a combination of single output GPs has the disadvantage of not modelling the coupling relationships among the outputs of a system as a ship. A ship is a system with highly related outputs where the absence of the relation between outputs can carry to error in prediction.

In the present study, non-parametric dynamic SI for ships is proposed with the use of multi-output GPs, NARX structure and gradient descent optimization. The output from the algorithm will be a predictive value and a measure of confidence of the predictive value. Multi-output GPs is a special case of GPs with the capability to model the nonlinear behaviour and coupling among outputs of a multi-output system. Ships are ideal candidates for the use of multi-output GPs owing to their dynamic system with highly coupled outputs, i.e. the ship's motion in 4 DoF. The present implementation was made over data obtained from a non-conventional zig-zag test with variable frequency of a 4 DoF simulated container ship. Multiple sample times and data length were tested to find the best metric that can describe a ship. In addition to the algorithm development, another immediate objective of the study is the demonstration of the viability of GPs in modelling ships.

2.2 Nonlinear Dynamic Ship Model

[19] proposed a 4 DoF (surge, sway, yaw and pitch) mathematical nonlinear model for ships including the contribution from hydrodynamics added masses. In respect to a

body fixed frame (Figure 2.1) the mathematical model can be expressed as:

$$\begin{aligned}
&(m + m_x)\,\dot{u} - (m + m_y)\,vr = X \\
&(m + m_y)\,v + (m + m_x)\,ur + m_y\alpha_y\dot{r} - m_y l_y\dot{p} = Y \\
&(I_x + J_x)\,\dot{p} - m_y l_y\dot{v} - m_x l_x ur = K - W\overline{GM_T}\phi \\
&(I_z + J_z)\,\dot{r} + m_y\alpha_y\dot{v} = N - x_G Y
\end{aligned} \tag{2.1}$$

where the added mass in x-axis and y-axis are represented by m_x,m_y and the added

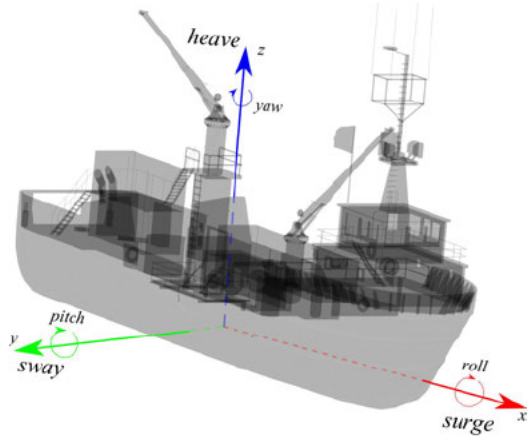

Figure 2.1: Definition of body fixed coordinated systems.

moment of inertia about x-axis and y-axis are represented by J_x and J_y. The centre of added mass is denoted by the vector

$$(\alpha_x, \alpha_y, \alpha_z)$$

, while the added mass centre for m_x and m_y is denoted by the z-coordinates of l_x and

l_y . The vector $[X, Y, K, N]$ expresses the forces over the vehicle and can be defined as:

$$\begin{aligned} X = X(u) + (1-t)T + X_{vr}vr + X_{vv}v^2 + X_{rr}r^2 \\ +X_{\phi\phi}\phi^2 + X_{\delta}\sin(\delta) + c_{RX}F_N\sin\delta \end{aligned}$$

$$\begin{aligned} Y = Y_v v + Y_r r + Y_\phi \phi + Y_p p + Y_{vvv}v^3 + Y_{rrr}r^3 \\ +Y_{vvr}v^2 r + Y_{vrr}vr^2 + Y_{vv\phi}v^2\phi + Y_{v\phi\phi}v\phi^2 \\ +Y_{rr\phi}r^2\phi + Y_{r\phi\phi}r\phi^2 + Y_\delta\cos(\delta) \\ -(1+a_H)\,z_R F_N\cos\delta \end{aligned}$$

$$\begin{aligned} K = K_v v + K_r r + K_\phi \phi + K_p p + K_{vvv}v^3 + K_{rrr}r^3 \\ +K_{vvr}v^2 r + K_{vrr}vr^2 + K_{vv\phi}v^2\phi + K_{v\phi\phi}v\phi^2 \\ +K_{rr\phi}r^2\phi + K_{r\phi\phi}r\phi^2 + K_\delta\cos(\delta) \\ +(1+a_H)\,F_N\cos\delta \end{aligned} \tag{2.2}$$

$$\begin{aligned} N = N_v v + N_r r + N_\phi \phi + N_p p + N_{vvv}v^3 + N_{rrr}r^3 \\ +N_{vvr}v^2 r + N_{vrr}vr^2 + N_{vv\phi}v^2\phi + N_{v\phi\phi}v\phi^2 \\ +N_{rr\phi}r^2\phi + N_{r\phi\phi}r\phi^2 + N_\delta\cos(\delta) \\ +(x_R + a_H x_H)\,z_R F_N\cos\delta \end{aligned}$$

where:

$X(u)$ = function dependent on the velocity $X_{|u|u}\,|u|\,u$

δ = rudder angle

F_N =rudder force

$$X_{vr}X_{vv}, ...N_{r\phi\phi}$$

=model parameters

As can be seen, the mathematical model is defined by more than 50 parameters including parameters from the actuation surfaces. An example of the hydrodynamic parameters and its application can be found in [21].

2.3 Dynamic Identification with Multi-output GPs

The design of the algorithm for multi-output SI with GPs is based on the previous work of[12]. The dynamic identification problem can be defined as the search for relation between a vector formed by delayed samples from the inputs $u(k)$ and outputs $y(k-1)$ and the future output values. The relationship can be expressed by the equation:

$$\mathbf{y}(k+1) = f\left(\mathbf{z}(k), \mathbf{\Theta}\right) + v(k) \tag{2.3}$$

where

$$f\left(\mathbf{z}(k), \mathbf{\Theta}\right)$$

is a function that maps the sample data vector

$$\mathbf{z}(k)$$

that contains the vector $[\mathbf{u}(k-1), \mathbf{y}(k-1)]$ to the output space based on the hyper-parameters $\mathbf{\Theta}$. $\mathbf{v}(k)$ accounts for the noise and error in the prediction of output $\mathbf{y}(k)$. In the case of dynamic SI, the discrete time variable (k) is presented as an embedded element in the regression process as it is accounted in the delayed samples.

A requirement for dynamic SI of nonlinear systems is the selection of a nonlinear model structure as nonlinear autoregressive model with exogenous input (NARX), nonlinear autoregressive (NAR), nonlinear output-error (NOE), nonlinear finite-impulse response (NFIR), etc. From all the possible structures, the simpler and most popular structure to implement is NARX as its predictions are based on previous measurements of the input signals and output signals and require a more simplified optimization scheme. In the case of a ship, NARX is the most practical configuration since the measuring points are restricted to the available sensors. Figure 2.2 shows the NARX configuration for Dynamic GPs for a simple case of one-input one-output system.

In the case of a single-input single-output structure NARX for a GPs, the inputs signals are not considered separately as they are grouped into a single vector of dimension n that derives to an output of single dimension. In the case of a four DoF ship, the system can be defined a function f who depends of a vector formed by the respective regressors

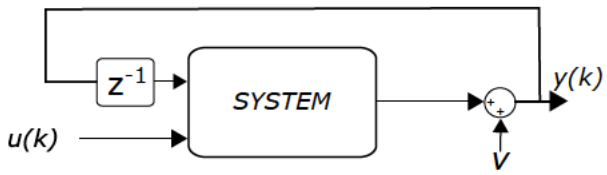

Figure 2.2: NARX for single input, single output system.

of each output and the regressors of the command signals of propeller and rudder such as.

$$\mathbf{y} = f\left(y_{(k-1:n)}, \mathbf{u}_{RPM\,(k-1:n)}, \mathbf{u}_{rudder\,(k-1:n)}\right) \tag{2.4}$$

If a Newton-Lagrange mathematical model had been used, our system will have two-input signals, four-output system signals. (Figure 2.3) presents the graphical representation of the NARX architecture used with multi-output GPs with four vector of dimension R3.

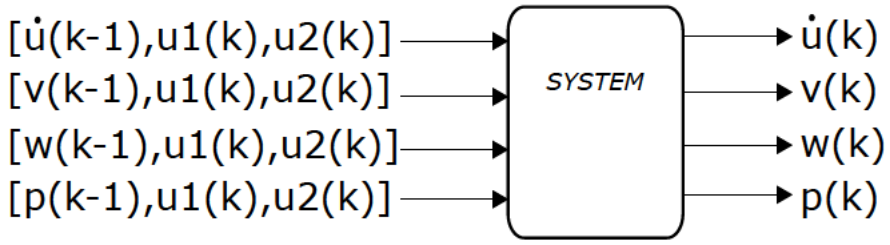

Figure 2.3: NARX structure for dynamic SI of nonlinear container ships. $u1$ is the measure RPM and $u2$ is the rudder angle at time k.

2.3.1 Multi-output GPs

The previous sections outline (equation (2.1)) and (equation (2.2)) which show the level of coupling between the Newton-Lagrange equations of a ship. The nonlinearity and coupling between outputs are better represented by a multi-output GPs. multi-output GPs presented here is based on the work of [39]. multi-output GPs are founded in the regression of data by the convolution of white noise process with a smoothing function[40]. This was later introduced by [41] to the machine learning community by assuming multiple latent process defined over a space $\Re^q$. The dependency between two outputs is modelled with a common latent process and their independency with a latent

function who does not interact with other outputs. If a set of functions $\{f_q(\mathbf{x})\}_{q=1}^{Q}$ is considered, where Q is the Output Dimension for a N number of data points, where each function is expressed as the convolution between a smoothing kernel $\{k_q(\mathbf{x})\}_{q=1}^{Q}$ and a latent function $\mathbf{u}(z)$,

$$f_q(x) = \int_{-\infty}^{\infty} k_q(\mathbf{x}-\mathbf{z})\, u(\mathbf{z})\, d\mathbf{z} \tag{2.5}$$

This equation can be generalized for more than one latent function $\{u_r(\mathbf{x})\}_{r=1}^{R}$ and include a corruption function (noise) independent to each of the outputs $w_q(\mathbf{x})$, to obtain

$$\begin{aligned} \mathbf{y}_q(\mathbf{x}) &= f_q(\mathbf{x}) + w_q(\mathbf{x}) \\ \mathbf{y}_q(\mathbf{x}) &= \sum_{r=1}^{R} \int_{-\infty}^{\infty} k_{qr}(\mathbf{x}-\mathbf{z}) u_r(\mathbf{z})\, d\mathbf{z} + w_q(\mathbf{x}) \end{aligned} \tag{2.6}$$

The covariance between two different functions $y_q(x)$ and $y_s(x')$ is:

$$\begin{aligned} \operatorname{cov}\left[\mathbf{y}_q(\mathbf{x}), \mathbf{y}_s(\mathbf{x}')\right] &= \operatorname{cov}\left[f_q(\mathbf{x}), f_s(\mathbf{x}')\right] \\ &+\operatorname{cov}\left[w_q(\mathbf{x}), w_s(\mathbf{x}')\right] \delta_{qs} \end{aligned} \tag{2.7}$$

where

$$\begin{aligned} \operatorname{cov}\left[f_q(\mathbf{x}), f_s(\mathbf{x}')\right] = \sum_{r=1}^{R} \sum_{p=1}^{R} \int_{-\infty}^{\infty} k_{qr}(\mathbf{x}-\mathbf{z}) \\ \int_{-\infty}^{\infty} k_{sp}(\mathbf{x}'-\mathbf{z}') \operatorname{cov}\left[u_r(\mathbf{z}), u_p(\mathbf{z}')\right] d\mathbf{z}' d\mathbf{z} \end{aligned} \tag{2.8}$$

If it is assumed that $u_r(\mathbf{z})$ is an independent white noise $\operatorname{cov}\left[u_r(\mathbf{z}), u_p(\mathbf{z}')\right] = \sigma_{ur}^2 \delta_{rp} \delta_{z,z'}$, equation (2.8) will become:

$$\operatorname{cov}\left[f_q(\mathbf{x}), f_s(\mathbf{x}')\right] = \sum_{r=1}^{R} \sigma_{ur}^2 \int_{-\infty}^{\infty} k_{qr}(\mathbf{x}-\mathbf{z}) k_{sp}(\mathbf{x}'-\mathbf{z}') d\mathbf{z} \tag{2.9}$$

The mean $\widehat{\mathbf{y}}'$ with variance $\sigma_{\widehat{\mathbf{y}}'}$ of a predictive distribution at the point$\mathbf{x}'$ given the hyperparameters $\boldsymbol{\Theta}$ can be defined as

$$\widehat{\mathbf{y}}' = k(\mathbf{x}', \mathbf{x}) k(\mathbf{x}, \mathbf{x})^{-1} \mathbf{y} \tag{2.10}$$

and variance

$$\sigma^2_{\widehat{y}'} = k(\mathbf{x}', \mathbf{x}') - k(\mathbf{x}', \mathbf{x})^T k(\mathbf{x}, \mathbf{x})^{-1} k(\mathbf{x}', \mathbf{x}) \tag{2.11}$$

A complete explanation over the convolution process can be found in [39] and a complete implementation in [42].

2.3.2 Learning Hyperparameters

There are two principal methods for learning the hyperparameters Θ, Bayesian model interference and marginal likelihood. Bayesian inference is based on the assumption that a prior data of the unknown function to be mapped is known. A posterior distribution over the function is refined by incorporation of observations. The marginal likelihood method is based on the aspect that some hyperparameters are going to be more noticeable. Over this base the posterior distribution of hyperparameters can be described with a unimodal narrow Gaussian distribution.

The learning of GPs hyperparameters Θ is commonly done by the maximization of the marginal likelihood. The marginal likelihood can be expressed as:

$$p\left(\mathbf{y} \,|\mathbf{x}, \boldsymbol{\Theta}\right) = \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{K}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{y}^T \mathbf{K}^{-1} \mathbf{y}} \tag{2.12}$$

where $\mathbf{K}$ is the covariance matrix, N is the number of input learning data points and y is a vector of learning output data of the form $[y_1; y_2; \cdots y_N]$. To reduce the calculation complexity, it is preferred to use the logarithmical marginal likelihood that is obtained by the application of logarithmic properties to 11.

$$\mathcal{L}\left(\boldsymbol{\Theta}\right) = -\frac{1}{2}\log\left(|\mathbf{K}|\right) - \frac{1}{2}\mathbf{y}^T \mathbf{K}^{-1} \mathbf{y} - \frac{N}{2}\log\left(2\pi\right) \tag{2.13}$$

To find a solution for the maximization of log-likelihood multiples methods of optimization can be applied, like, particle swarm optimization, genetic algorithms, or gradient descent. For deterministic optimization methods, the computation of likelihood partial

derivatives with respect to each hyperparameter is require. From [43] log-likelihood derivatives for each hyperparameter can be calculated by:

$$\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{\Theta}_i} = -\frac{1}{2} trace\left(\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \mathbf{\Theta}_i}\right) + \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \mathbf{\Theta}_i}\mathbf{K}^{-1} \tag{2.14}$$

Equation (2.13) gives us the learning process computational complexity, for each cycle the inverse of the covariance matrix of $\mathbf{K}$ has to be calculated. This calculation carries a complexity $\mathbf{O}(\mathbf{NM})^3$ where N is the number of data points and M is the number of outputs of the system. After learning, the complexity of predicting the value $\mathbf{y}(k+1)$ is $\mathbf{O}(\mathbf{NM})$ and to predict the mean value $\sigma(k+1)$ is $\mathbf{O}(\mathbf{NM})^2$. The higher training complexity $\mathbf{O}(\mathbf{NM})^3$ is the major disadvantage of using multi-output GPs. If the number of data increases the complexity of learning the hyperparameters increases in a cubic form. Methods such as genetic algorithms, differential equations, and particle swarm optimization can be applied to avoid the calculation of the marginal likelihood partial derivatives and thereby reduce the computational time.

2.4 Experiment Setup and Results

2.4.1 Experiment setup

The implementation of [19] mathematical model of a container ship programmed in the Marine Systems Simulator (Fossen and Perez, 2004) was used to create the required databases. The container ship particulars can be found in Table 1. A simulation setup was developed in MATLAB/Simulink to emulate the behaviour of a container ship (Figure 2.4). 1400 seconds were simulated where the inputs signals are constant shaft speed in RPM and a cosine signal with frequency change for rudder angle in radians (Figure 2.5). The objective of not using a standard test as zigzag or turning circle is to test the ability of GPs for online learning. A sample data point was captured for each three steps over the input and outputs. A total of 1868 points were captured over four outputs and 934 point over two input signals. The data set was divided in two sets of points, the first set of points is used for the model learning, and the second set of points is used for learning validation. The Validation data is purposely chosen to be

Table 2.1: Particulars of container ship

Parameter	**Magnitude**
Length overall	175 m
Breadth	25.4 m
Max. Rudder Angle	10 deg
Max. shaft velocity	160 RPM
Displacement Volume	21222 m^3
Rudder Area	33.0376 m^2
Propeller diameter	6.533 m

beyond the range of training data to test the ability of the method to predict beyond the training range. Two neural network nonlinear system identification models were also prepared. The first system (RNN1) was a recurrent neural network system and it has a similar architecture to the Multi-output GPs $f\left[\mathbf{y}_{(k-1)}, \mathbf{u}_{RPM\,(k-1:2)}, \mathbf{u}_{rudder\,(k-1:2)}\right]$ for each output. The second NN system (NN2) use a common NARX identification methodology and used the last four delayed outputs of the system and the last delayed input commands $f\left[\mathbf{y}_{(k-1:4)}, \mathbf{u_1}_{(k-1:2)}, \mathbf{u}_{2(k-1:2)}\right]$ for each output. The neural network systems use a Log-sigmoid transfer function, at different of GPs the training of NN was done by Levenberg-Marquardt backpropagation. Both neural network systems were trained, validated, and tested with the same data used for the multi-output GPs. The complete implementation code can be found at the GitHub Repository [1].

2.4.2 Training and validation

The software written by [42] was softly modified to accept the multidimensional input vectors and a script was written to implement the NARX structure. The convolution of two square exponential Gaussian processes and a white noise was chosen as kernel. The inputs of the GPs were defined as four inputs of dimension five of the form:

[1] https://github.com/ArizaWilmerUTAS/Multi-Output-GPs-Identification-SHIPfghdasDDXSZAQASZdfghwertyu

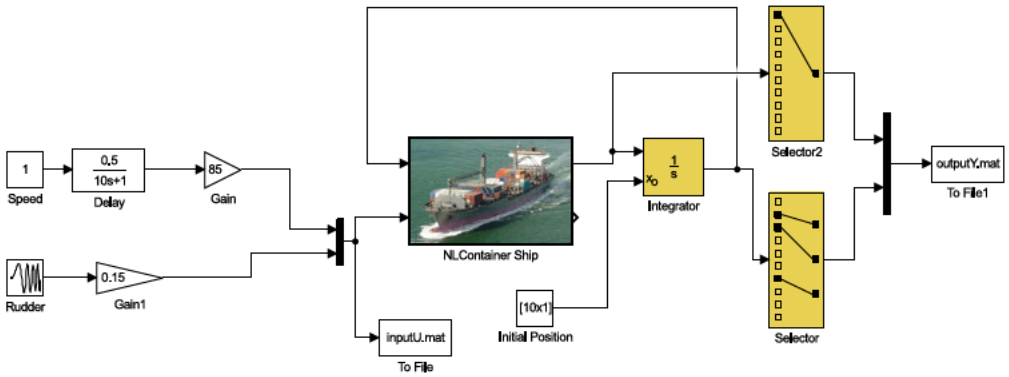

Figure 2.4: Simulink Container Ship Implementation

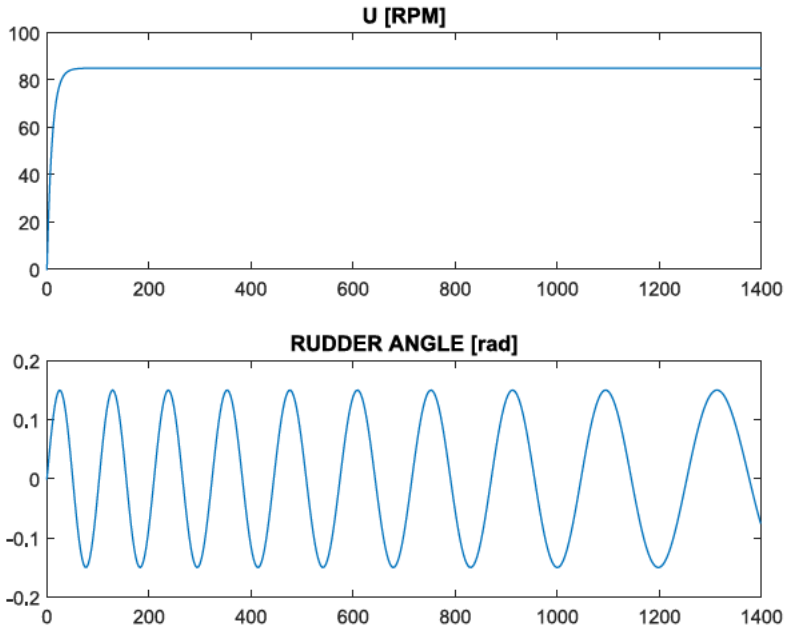

Figure 2.5: Shaft speed [rpm] and rudder angle signals for simulation of Ship.

$$\begin{bmatrix} \dot{u}_k \\ v_k \\ r_k \\ p_k \end{bmatrix} = f\left(\mathbf{y}_{k-1}, \mathbf{u}_{RPM\,(k-1:2)}, \mathbf{u}_{rudder\,(k-1:2)}\right) \tag{2.15}$$

where $\mathbf{y}_{k-1}$ is the first regressor of the output vector $[\dot{u}_k, v_k, r_k, p_k]$.

The selection of the structure of regressors was determined via the examination of the

mathematical model. Each output is affected by the past states of output and rudder force F_N produced by the interaction of the rudder angle and the propeller RPM as both signals are required for the calculation of F_N . Under this assumption different structures were tested to verify the responsiveness to each regressor. The test showed that the container ship system is more responsive to regressors from the rudder angle and the propeller RPM.

The captured output vector was the derivative of surge speed, the speed in sway and the angular speeds of yaw and roll, $[\dot{u}, v, r, p]$. As can be seen in equation (2.1) and equation (2.2) the surge speed is not highly couple to the other system outputs, in our simulation capturing the surge speed and posterior simulation was not converging to the real output, in contrast the surge speed derivative shows coupling with other system outputs. The input signals and outputs were normalized between -1 and 1 to give all the inputs and outputs the same weight in the learning process.

For the training, the minimization of the negative logarithmical likelihood was used along with the scaled conjugate gradient with multiple start points to insure convergence. Figure 2.6 shows the results of GPs training compared to the real system signals, and the error plots between the predicted and real systems. In all the graphs, a confidence band 2σ is plotted. The error for the surge derivative is less than 0.02 over the training data.

The validation data consisted of the real output from the training data with the system delay $(k-1)$ in vector form with the delayed commanded inputs. The segments of results from the validation with the second set of data are depicted in Figure 2.7, the predicted output and confidence of 2σ band is portrayed in comparison to the original system. The low validation errors show a good system prediction for the sway speed and yaw speed. It can be notice that the simulation precision is lose by how far from the training data the step is. The variance in our validation results increase as the data used for validation drift away from the trained operational region. This was done with the objective to test the capability of GPs to predict outside the trained operational region.

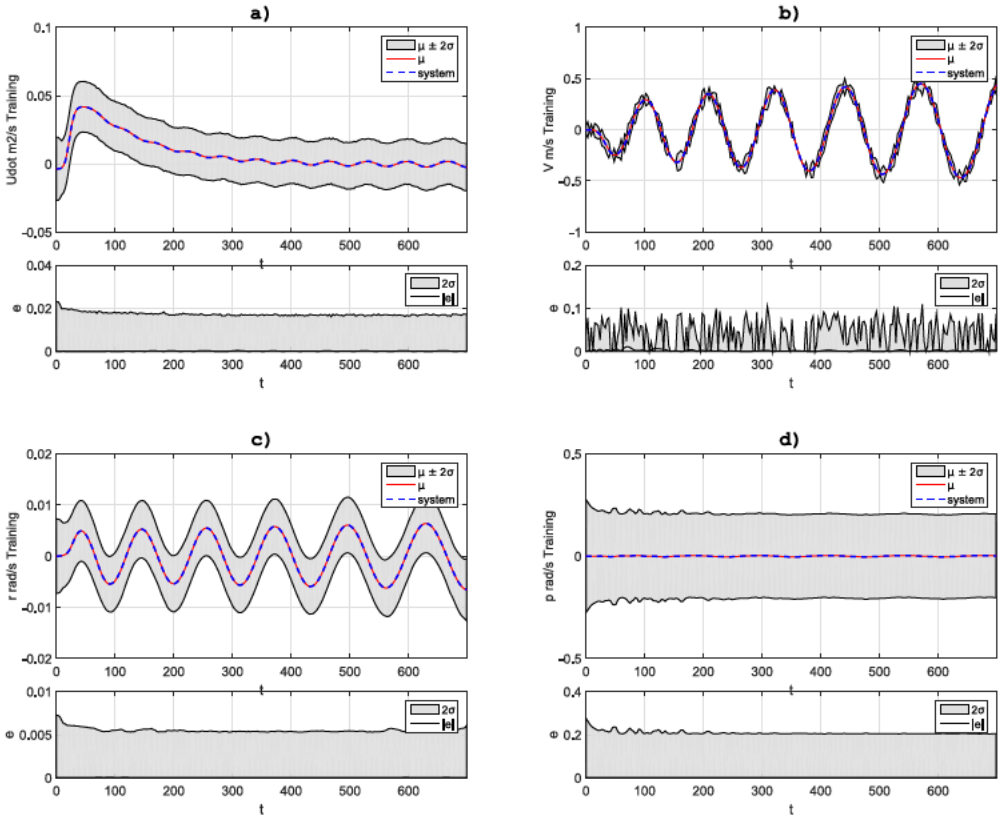

Figure 2.6: Prediction from Multioutput-GPs obtained model with training data (0-700 seconds) compared to mathematical model, a) controlled surge acceleration, b) induced sway speed, c) controlled yaw speed, and d) induced roll speed.

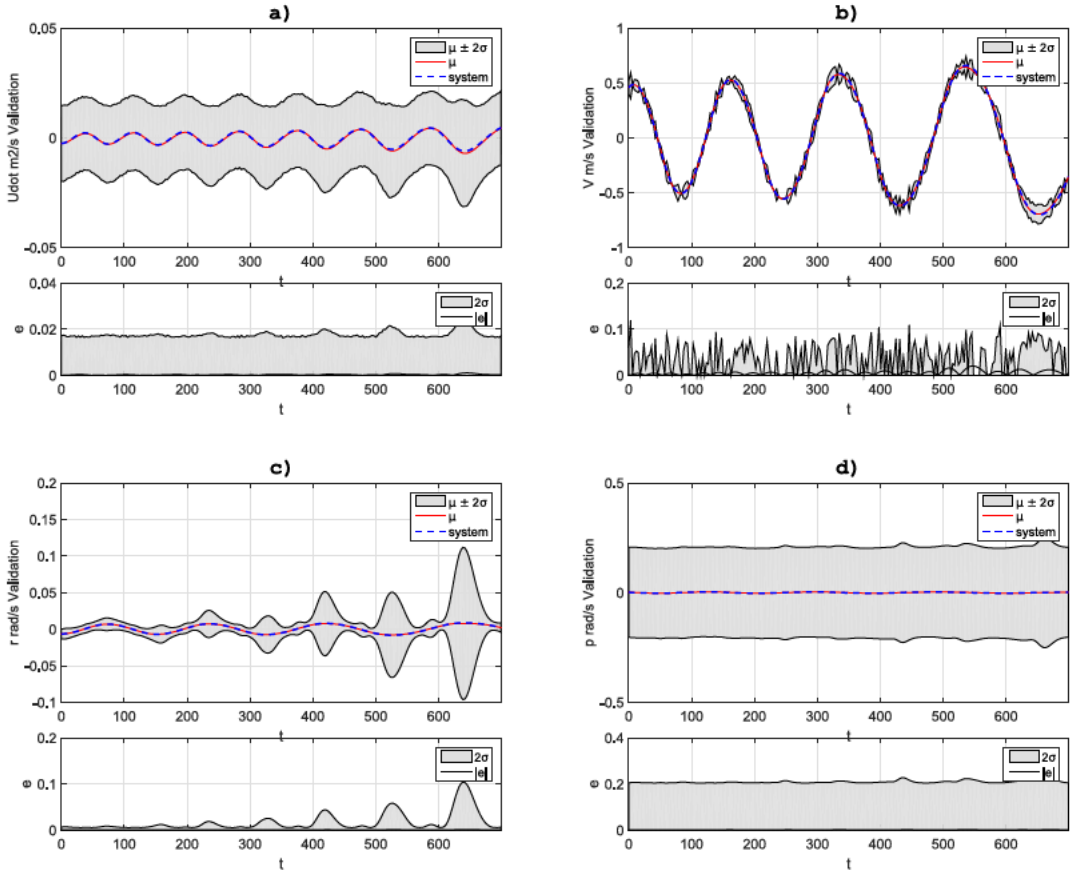

Figure 2.7: Prediction from Multioutput-GPs obtained model with validation data (700-1400 seconds) compared to mathematical model, a) controlled surge acceleration, b) induced sway speed, c) controlled yaw speed, and d) induced roll speed.

2.4.3 Simulation

A third step was implemented in the way of a naive simulation. Methods of control with non-parametric models require a number of step forward of prediction to be able to control a system. With the objective of testing the ability to predict a system from past data, a naive simulation was setup. At each step the output from the simulation is feedback to the simulation as the past input $y_i(k-1)$, the initial position and control signal of rudder and forward speed where used, the naive simulation covers training(0-700s) and validation data(701-1400s) acquired from the original simulation. Table 2 shows the root mean square error (RMSE), the predicted residual error sum of squares (PRESS) measurements for the simulation stage over the training and validation data, and the training time and step simulation time for each of the methodologies. The RMSE and PRESS value for the proposed GPs are smaller than the other systems. As evident in Figure 2.8(a-c) NNARX system with the same architecture (marked as NarxNN) and data as in the multi output GPs has limitations in the capability to predict the system behaviour beyond the training range in all DoF. The more complex RNN system (RNN1) produces relatively good results, except in predicting the surge. This is evident Figure 2.8 (a) where RNN1 results in large deviations from the original system, especially after 1000s.The yaw output in Figure 2.8(c) shows a higher variance as results of higher association to the other outputs of the system and similitude to other training data this is because of normalization of the outputs in the training data. The difference in capability of prediction of the system is related to their internal functions and how they relate the training data. In comparison to NNRX and RNN, the multi-output GPs show similar performance than RNN outside the training horizon in all the DoF. This is evident in all the results shown in Figure 2.8 with the close match to the system from the simulation, it can be established that the Gaussian model can be used for applications as control and failure detection as it can predict future system states with the added value of a confidence measure.

Table 2.2: Summary prediction quality measurements

	GPs	NarxNN	RNN1
RMSE	0.0091	0.0092	0.044
PRESS	0.2337	5.47	0.2382
Training time(s)	779	245	125
Step simulation time	0.0625	0.032	0.027

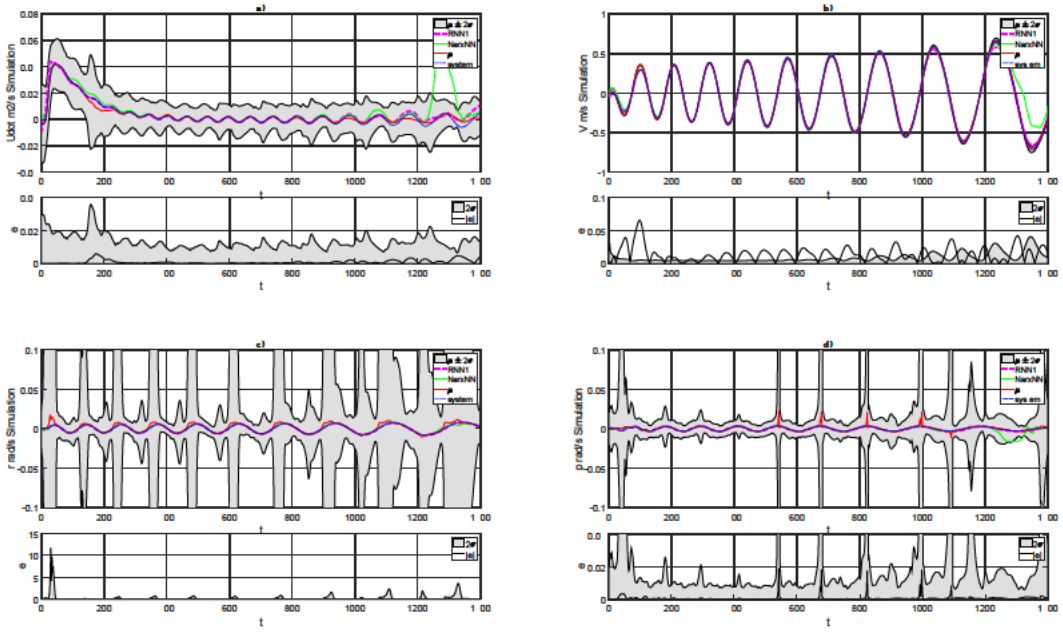

Figure 2.8: Prediction from Multi-output GPs by algorithm of naive simulation with full data from input signals compared to mathematical model, a) controlled surge acceleration, b) induced sway speed, c) controlled yaw speed, and d) induced roll speed.

2.5 Conclusion

The basic methodology for the use of multiple-output Gaussian distribution for the identification of ships dynamical models is presented in this paper. The methodology has been validated with the data obtained from a coupled dynamical system of a container ship. With the proposed Gaussian model, the large number of system parameters found in a typical ship model can be reduced to a smaller number of hyperparameters. A standard validation process of machine learning and prediction over the complete data set of training and validation were executed to establish the model quality and robustness of the algorithm. The prediction of the full set of data based in a start value and feedback from the last prediction step show low error. As the results indicate, multi-output GPs has the ability to model complex dynamic system having highly coupled outputs and provide a measure of the confidence represented by the variance.

The use of other methods such as sparse multi-output GPs and the use of more powerful prediction techniques as Taylor series or Montecarlo method can take advantage of the variance to increase the horizon of cover manoeuvres and the prediction accuracy.Although the results obtained look encouraging, conclusion about the practical value of the method can only be obtained by comparison with other GPs methods and validation with real data from a ship or other oceanic vehicle.

Chapter 3

Non-parametric Dynamic System Identification of Underwater Vehicles Using Multi-Output Gaussian Processes

The present chapter advance over the results obtained in chapter 3 to propose a methodology for system identification of underactuated AUV. At the time of submission of this thesis, the present chapter is under review by the *Journal of Transactions of the Institute of Measurement and Control*.

3.1 Introduction

Dynamic modelling of unmanned underwater vehicles (UUVs) has been a subject of interest among researchers since the early days of underwater exploration. Nowadays, UUVs are extensively employed in research, industry and military applications. Modelling of autonomous underwater vehicles (AUVs) is an important step for design of mission, control and navigation systems. Thus, accurate modelling and adaptability of such systems is an important issue due to such extensive applications. The most common methodologies use a mathematical model which is derived from Newtonian-Lagrange mechanics. This mathematical model is composed of a series of coefficients that need to be calculated to obtain an accurate model. Difference between the obtained model and the reality is usually treated in the literature as noise and in most cases, is modelled as a Gaussian distribution. A Gaussian distribution can be extended to the calculation of an approximation to a real model of a vehicle with higher exactitude and adaptability than a mathematical model[12].

Over the years, multiple methods for the calculation of coefficients of underwater vehicles have been proposed. One way to obtain the hydrodynamic coefficients is to perform a series of captive model tests such as rotating arm and planar motion mechanics [44–46]. Another common technique for the hydrodynamic coefficient calculation is the use of computational fluid dynamics (CFD) [47]. However, for the successful application, CFD still requires verification of results with experiments [48].

Nevertheless, research has probed the variability of mathematical models for AUVs as the vehicle operates in proximity to objects [49], near surface [50] and most commercial available underwater vehicles with a modular architecture involving variable geometric and mass. Furthermore, in certain applications, the precision of some coefficients is require to be within 5

Another procedure to obtain coefficients from a model of an underwater vehicle is the use of observers. Common algorithms applied to obtain the hydrodynamic coefficients of AUVs from measured data are least-squares [51, 52], nonlinear Kalman filters such as extended (EKF) and unscented Kalman filter (UKF) [53]. The EKF requires the linearization at each time step for the approximation of non-linearities which can be

difficult to regulate and implement. A method to overcome this is the use of UKF which applies the unscented transform over a set of methodically chosen samples to model the system nonlinearity [54]. Other common methodologies are frequency domain identification [55]), neural networks (NN) [56] and support vector machines (SVM) [57]. The latter two methodologies are machine-learning algorithms and are more commonly used in online learning of the coefficients and also provide system adaptability. The adaptation of mathematical model has inherent defects such as the dependency of initial values, small quantity of coefficients to be updated, ill-conditioned matrix and drift.

Machine learning algorithms are not limited to the calculation of hydrodynamic coefficients as they can learn to behave as part of the system or the complete system. Multiple applications have taken advantage of this ability and used NN [58] and SVM [59] to learn the damping model for the system which is placed in parallel to a well-known partial mathematical model. Other applications have used pure machine learning algorithms to identify a complete underwater vehicle as a black-box model with the use of nonlinear autoregressive model with exogenous (NARX) architecture. [60] have used multiple architectures of NN for the regression of an AUV model in a NARX architecture and used the learned model for model predictive control. Their study shows that recurrent NN (RNN) provides higher faithfulness to the plant.

Recently, [61] compared different machine learning algorithms for the system regression of underwater vehicles, i.e. NN, SVM, Gaussian Process Regression (GPR) and Kernel Ridge regression (KRR). Their results show that the machine learning algorithms could model an AUV from onboard sensor data in comparison to a least squares approach. Nevertheless, in their study, a structure for dynamic system identification has not been employed and each degree of freedom was treated as separate element. This can be problematic in AUVs as the outputs are strongly coupled. In the specific case of modelling with Gaussian Processes (GPs) [12], research shows that the dynamic regression of a system with GPs can produce better results than other methodologies. The most common methodology for Multi-Input-Multi-Output systems is to model each DoF as a separate system [62]. More advance methodologies for Dynamic system identification have been proposed in [63] and [64] and specific methodologies are introduced for the identification of multi-output GPs based on the use of variation of dependent GPs.

GPs is a well-established methodology in fields such as geostatistics, where the method is called 'kriging' [36]. In GPs-based system identification, the model is built over input-output data and a covariance function is used to characterize the vehicle behaviour. The advantage of GPs is their ability to work with small quantities of data and noisy data. The predicted results consist of a mean and variance value which can be used for other purposes as well such as control, navigation and model based fault detection as it contains a measure of confidence.

Multi-output GPs are a special case of GPs with the capability to model the nonlinear behavior and coupling among outputs of a multi-output system [65], both which are important for AUV dynamics. In this study, a non-parametric dynamic system identification with Multi-Output GPs architecture employed by the authors for ships [13] is extended to AUVs. The output from the algorithm will be a predictive value and a measure of confidence of the predictive value. The present implementation was made over data obtained from a non-conventional test with variable frequency of a nonlinear simulation model of a REMUS 100 AUV. Multiple sample times and data length were tested to find the best metric that can describe an AUV. A RNN was employed as a comparison to measure the effectiveness of the proposed method.

3.2 Nonlinear Dynamic AUV Model

In [21] it was shown that the nonlinear dynamic equations of motion of an underwater vehicle can be expressed in vector notation defined by a state vector composed by the vector v of velocities on the body frame of the form $[u, v, w, p, q, r]^T$ and the vector η of position in the Earth fixed frame(Figure 3.1) of the form $[\xi, \eta, \zeta, \phi, \theta, \psi]^T$ such that

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\,\mathbf{v} + \mathbf{D}(\mathbf{v})\,\mathbf{v} + \mathbf{g}(\eta) = \tau \tag{3.1}$$

with the kinematic equation

$$\dot{\eta} = \mathbf{J}(\eta)\,\mathbf{v} \tag{3.2}$$

where

η position and orientation of the vehicle in Earth-fixed frame,

$\mathbf{v}$ linear and angular vehicle velocity in body fixed frame,

$\dot{\mathbf{v}}$ linear and angular vehicle acceleration in body fixed frame,

$\mathbf{M}$ matrix of inertial terms,

$\mathbf{C}(\mathbf{v})$ matrix of Coriolis and centripetal terms,

$\mathbf{D}(\mathbf{v})$ matrix consisting of damping or drag terms

$\mathbf{g}(\eta)$ vector of restoring forces and moments due to gravity and buoyancy

τ vector of control and external forces

$\mathbf{J}(\eta)$ rotation matrix that converts velocity in a body fixed frame v to an Earth fixed frame velocity $\dot{\eta}$.

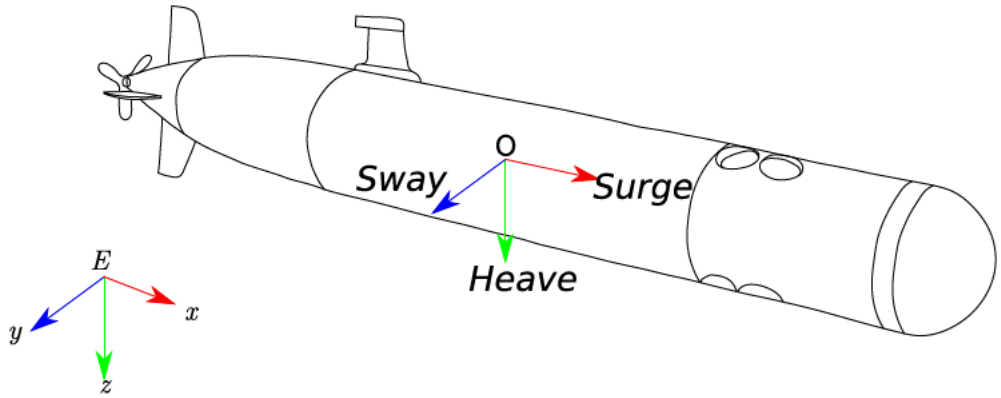

Figure 3.1: AUV different reference frames, vehicle frame is equal to centre of buoyancy.

Equation (3.1) can be expanded into a more general equation of motion as has been shown in [66, 67].The result of the expansion will be a system of six equation with 73 hydrodynamic coefficients. However, for a complete model the control surfaces must be modelled. In a general case, the resulting forces and moments of a control surface (thrusters and fins) can be expressed as [21]

$$\begin{aligned} F_{prop} &= -K_{fprop}\,|n|\,n \\ M_{prop} &= -K_{mprop}\,|n|\,n \end{aligned} \tag{3.3}$$

$$\begin{aligned} L_{fin} &= K_{L|\delta_{\text{fin}}}\delta_{\text{fin}} v_e^2 \\ M_{fin} &= K_{M|\delta_{\text{fin}}}\delta_{\text{fin}} v_e^2 \end{aligned} \tag{3.4}$$

A more accurate thruster model can be found in [68] with the inclusion of the motor model and fluid dynamics. However, in this study, a more conservative model is used. Details of the Remus 100 AUV model used in this study are given in Section III.

3.3 Dynamic Identification with Multi-output GPs

GPs can be defined as a generalization of a multivariate Gaussian distribution. A multivariate Gaussian distribution is defined by it's mean and a covariance matrix. In the case of GPs is a distribution over functions rather a distribution over vectors. GPs is one of the methods based on kernel functions where the kernel function calculates the relationship between an input and an output point, and generates the covariance between them. The covariance determines how strongly linked (correlated) these two points are. In the case of multi-output GPs this is extended by the convolution of kernels to add not only the relationship between an input and an output but also the relationship between the outputs. The kernel is the key ingredient for the calculation of the covariance matrix that correlates inputs and outputs of training data [69].

The design of the algorithm for multi-output system identification with GPs is based on the previous work of Alvarez and Lawrence [39] and [12]. The dynamic identification problem can be defined as the search for relationship between a vector formed by delayed samples from the inputs $\mathbf{u}(k-1)$ and outputs $\mathbf{y}(k-1)$ and the future output values. The relationship can be expressed by the equation:

$$\mathbf{y}(k+1) = f\left(\mathbf{x}(k), \Theta\right) + \mathbf{v}(k) \tag{3.5}$$

where $f\left(\mathbf{x}(k), \Theta\right)$ is a function that maps the sample data vector $\mathbf{x}(k)$ to the output space based on the hyperparameters Θ; $\mathbf{v}(k)$ accounts for the noise and error in the prediction of output $\mathbf{y}(k)$. In the case of dynamic system identification the discrete time variable (k) is presented as an embedded element in the regression process as it is accounted in the delayed samples.

A requirement for dynamic system identification of nonlinear systems is the selection of a nonlinear model structure such as nonlinear autoregressive model with exogenous input (NARX), nonlinear autoregressive (NAR), nonlinear output-error (NOE), nonlinear

finite-impulse response (NFIR) and other structures. From all the possible structures, the simpler and most popular structure to implement is NARX as it only requires measurements of system output/s and input/s. In the case of an AUV, NARX is the most practical configuration since the measuring points are restricted to the available sensors [12].

3.3.1 Multi-output GPs

The non-linear dynamic system of an AUV (equation (3.1)) shows the level of coupling between the Newton-Lagrange equations of an AUV. The nonlinearity and coupling between outputs can be better represented by a multi-output GPs. Multi-output GPs presented here are based on the work of [39]. Multi-output GPs are founded in the regression of data using convolving white noise process with a smoothing kernel function [40]. This was later introduced by [41] to the machine learning community by assuming multiple latent process defined over a space $\Re^q$. The dependency between two outputs is the model with a common latent process and their independency with a latent function, which does not interact with other outputs. If a set of functions $\{f_q(\mathbf{x})\}_{q=1}^{Q}$ is considered, where Q is the Output Dimension for a N number of data points, where each function is expressed as the convolution between a smoothing kernel $\{k_q(\mathbf{x})\}_{q=1}^{Q}$ and a latent function $u(\mathbf{z})$,

$$f_q(\mathbf{x}) = \int_{-\infty}^{\infty} k_q(\mathbf{x}-\mathbf{z})\, u(\mathbf{z})\, dz \tag{3.6}$$

This equation can be generalized for more than one latent function $\{u_r(\mathbf{x})\}_{r=1}^{R}$ and includes a corruption function (noise) independent to each of the outputs

$$w_q(\mathbf{x})$$

, to obtain

$$\begin{aligned} \mathbf{y}_q(x) &= f_q(\mathbf{x}) + w_q(\mathbf{x}) \\ \mathbf{y}_q(x) &= \sum_{r=1}^{R} \int_{-\infty}^{\infty} k_{qr}(\mathbf{x}-\mathbf{z}) u_r(\mathbf{z})\, d\mathbf{z} + w_q(\mathbf{x}) \end{aligned} \tag{3.7}$$

The covariance between two different functions $\mathbf{y}_q(\mathbf{x})$and $y_s(\mathbf{x}')$ is:

$$\begin{aligned} \operatorname{cov}\left[\mathbf{y}_q(\mathbf{x}), \mathbf{y}_s(\mathbf{x}')\right] &= \operatorname{cov}\left[f_q(\mathbf{x}), f_s(\mathbf{x}')\right] \\ &+ \operatorname{cov}\left[w_q(\mathbf{x}), w_s(\mathbf{x}')\right] \delta_{qs} \end{aligned} \tag{3.8}$$

where

$$\begin{aligned}\operatorname{cov}\left[f_q\left(\mathbf{x}\right), f_s(\mathbf{x}')\right] = \sum_{r=1}^{R}\sum_{p=1}^{R}\int_{-\infty}^{\infty} k_{qr}(\mathbf{x}-\mathbf{z}) \\ \int_{-\infty}^{\infty} k_{sp}(\mathbf{x}'-\mathbf{z}')\operatorname{cov}\left[u_r\left(\mathbf{z}\right), u_p(\mathbf{z}')\right] dz'dz\end{aligned} \tag{3.9}$$

If it is assumed that $u_r\left(\mathbf{z}\right)$ is an independent white noise $\operatorname{cov}\left[u_r\left(\mathbf{z}\right), u_p(\mathbf{z}')\right] = \sigma_{ur}^2\delta_{rp}\delta_{z,z'}$ will become:

$$\operatorname{cov}\left[f_q\left(\mathbf{x}\right), f_s(\mathbf{x}')\right] = \sum_{r=1}^{R}\sigma_{ur}^2\int_{-\infty}^{\infty} k_{qr}(\mathbf{x}-\mathbf{z})k_{sp}(\mathbf{x}'-\mathbf{z}')d\mathbf{z} \tag{3.10}$$

The mean $\widehat{\mathbf{y}}'$ with variance $\sigma_{\widehat{\mathbf{y}}'}$ of a predictive distribution at the point $\mathbf{x}'$ given the hyperparameters $\boldsymbol{\Theta}$ can be defined as

$$\widehat{\mathbf{y}}' = \mathbf{k}(\mathbf{x}',\mathbf{x})\mathbf{k}(\mathbf{x},\mathbf{x})^{-\mathbf{1}}\mathbf{y} \tag{3.11}$$

and variance

$$\sigma^{\mathbf{2}}_{\widehat{\mathbf{y}}'} = \mathbf{k}(\mathbf{x}',\mathbf{x}') - \mathbf{k}(\mathbf{x}',\mathbf{x})^{\mathbf{T}}\mathbf{k}(\mathbf{x},\mathbf{x})^{-\mathbf{1}}\mathbf{k}(\mathbf{x}',\mathbf{x}) \tag{3.12}$$

A comprehensive description and implementation of the convolution process can be found in [39] and [42] respectively. In this study, the convolution of two square exponential kernels are used since squared exponential kernel is a universal kernel [70], provided that data is stationary and the function to be modelled is a smooth one. Furthermore, squared exponential kernel has small quantity of hyperparameters to be established.

3.3.2 Learning Hyperparameters

There are two principal methods for learning the hyperparameters Θ namely: Bayesian model interference and marginal likelihood. Bayesian inference assumes that prior data of the unknown function to be mapped are known and a posterior distribution over the function is refined by incorporation of observations. The marginal likelihood method is based on the aspect that some hyperparameters are going to be more noticeable. Over this base, the posterior distribution of hyperparameters can be described with a unimodal narrow Gaussian distribution.

The learning of GPs hyperparameters Θ is normally carried out with the maximization of marginal likelihood. The marginal likelihood can be expressed as:

$$p\left(\mathbf{y}\left|\mathbf{x},\boldsymbol{\Theta}\right.\right) = \frac{1}{(2\pi)^{\frac{N}{2}}\left|\mathbf{K}\right|^{\frac{1}{2}}}e^{-\frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y}} \tag{3.13}$$

where $\mathbf{K}$ is the covariance matrix, N is the number of input learning data points and $\mathbf{y}$ is a vector of learning output data of the form $[y_1; y_2; \cdots y_N]$. To reduce the calculation complexity, it is preferred to use the logarithmical marginal likelihood that is obtained by the application of logarithmic properties to equation (3.14).

$$\mathcal{L}(\mathbf{\Theta}) = -\frac{1}{2}\log(|\mathbf{K}|) - \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{N}{2}\log(2\pi) \tag{3.14}$$

In order to find out a solution for the maximization of log-likelihood, there are multiple optimization methods that can be used such as particle swarm optimization, genetic algorithms, or gradient descent. For deterministic optimization methods, the computation of likelihood partial derivatives with respect to each hyperparameter is required. From [71] log-likelihood derivatives for each hyperparameter can be calculated by:

$$\frac{\partial\mathcal{L}(\mathbf{\Theta})}{\partial\mathbf{\Theta}_i} = -\frac{1}{2}trace\left(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\right) + \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\mathbf{K}^{-1} \tag{3.15}$$

Equation (3.14) gives us the learning process computational complexity, for each cycle the inverse of the covariance matrix of $\mathbf{K}$ has to be calculated. This calculation carries a complexity $\mathbf{O}(\mathbf{NM})^3$. After learning, the complexity of predicting the value $\mathbf{y}(k+1)$ is $\mathbf{O}(\mathbf{NM})$ and to predict the mean value $\sigma(k+1)$ is $\mathbf{O}(\mathbf{NM})^2$. The higher order term $\mathbf{O}(\mathbf{NM})^{\mathbf{3}}$ is the major disadvantage of using multi-output GPs. If the number of data increases the complexity of learning the hyperparameters increases in a cubic form. Methods such as genetic algorithms, differential equations, and particle swarm optimization can be applied to avoid the calculation of the marginal likelihood partial derivatives and thereby reduce the computational time.

3.4 Experiment Setup and Results

3.4.1 Experiment setup

The implementation of a mathematical model of an underactuated REMUS 100 AUV was used to generate the required identification data. The coefficients of [66] were used and adapted for simulation on Simulink with the addition of the thruster model from [72]. The AUV details can be found in Table 3.1. As shown in Figure 3.2 a simulation setup was developed in MATLAB/Simulink to emulate the AUV behavior. Figure 3.3

Table 3.1: REMUS 100 general characteristics.

Parameter	Value
Weight	299(N)
Buoyancy	306(N)
Vehicle total length	1.33(m)
Diameter	0.191
Max. Depth	100(m)

shows the input signals given for the rudder angle, thruster RPMs and elevator angle respectively. Simulation was carried out for 800 seconds. The objective of not using a standard test such as zigzag test or turning circle test is to test the ability of GPs under more drastic conditions. A sample data point was captured for each 1.5 seconds over the input and outputs. A total of 3200 points were captured over the six motion outputs and 1600 point over the three input signals (Propeller RPM, rudder angle and elevator angle). The data set was divided into two sets of points, the first set of points is used for the model learning, and the second set of points is used for learning validation. The validation data is purposely chosen to be beyond the range of training data to test the ability of the method to predict beyond the training range.

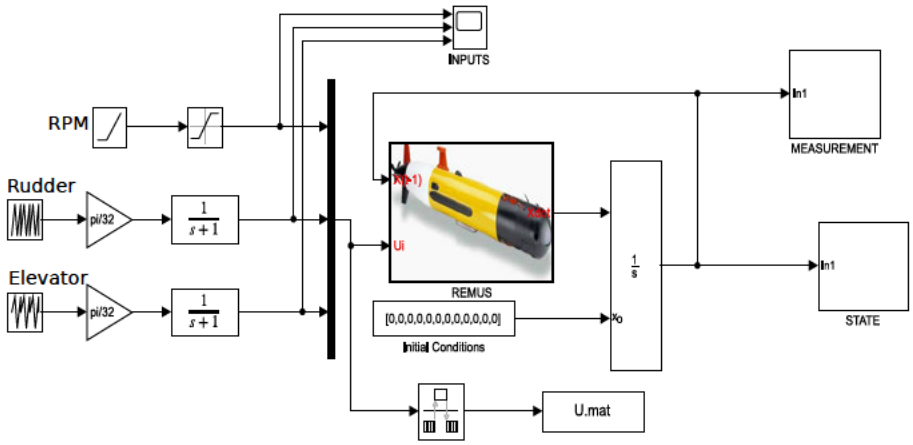

Figure 3.2: REMUS 100 AUV Simulink model.

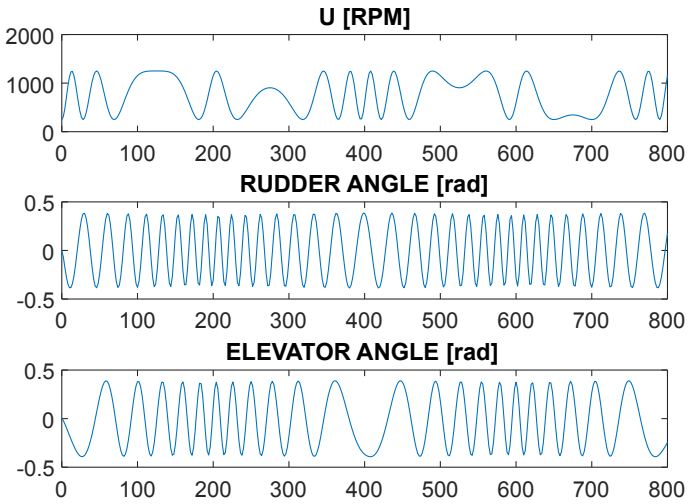

Figure 3.3: Remus 100 input signals.

3.4.2 Training and validation

A script was written to implement the NARX structure with multi-output GPs and the implementation of multi-out GPs by [42] was employed. The multi-output GPs regressors were defined as:

$$\begin{bmatrix} \mathbf{u}_k \\ \mathbf{v}_k \\ \mathbf{w}_k \\ \mathbf{p}_k \\ \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} = f\left(\mathbf{y}, \mathbf{c}_{k-1:3}\right) \tag{3.16}$$

Where $\mathbf{y}$ is the vector of regressors $[\mathbf{u}_{k-1:3}, \mathbf{v}_{k-1:3}, \mathbf{w}_{k-1:3}, \mathbf{p}_{k-1:3}, \mathbf{q}_{k-1:3}, \mathbf{r}_{k-1:3}]^T$, the function f is a relation between the vector of regressors from the correspondent vehicle speeds $(\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{p}, \mathbf{q}, \mathbf{r})$ or the full vehicle state $\mathbf{y}$, and the vector $\mathbf{c}$that content the regressors of the commanded signals $(u_{rpm}, u_{elevator}, u_{rudder})$ to the respective output of the system. The input signals were normalized between -1 and 1 to give all the inputs and outputs the same weight in the learning process.

For the training, a minimum search with the gradient descent method, in particular the

interior-point algorithm, was used for the minimization of the negative logarithmical likelihood.

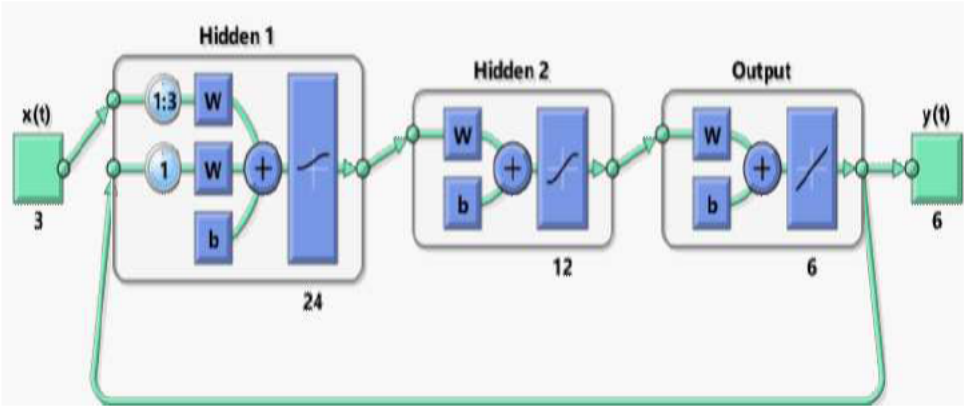

Figure 3.4: RNN configuration for AUV identification.

A neural network nonlinear system identification model for a comparison was also prepared. The NN system is a recurrent neural network (RNN) as shown in Figure 3.4. The MIMO RNN was setup with three terms of delays for the output to be feed back to the network and three terms delay of the inputs. The RNN that was selected as relatively optimal for the task at hand used two hidden layers with logarithmic sigmoid functions for the hidden layer neurons and was trained with Levenberg-Marquardt back-propagation, this configuration and regressors was selected as provide the best results for our system. A third step of simulation was carry out with the combination the full length of the data and feeding back after each step the output from the models. The neural network system was trained, validated, and tested with the same data used for the multi-output GPs. The complete implementation code can be found at the GitHub Repository [1]. Figure 3.5 presents the results of GPs training compared to the AUV simulator signals, and the error plots between the predicted and real systems. In all the figures, a 2σ variance is plotted. The variance values of the training data are in the

[1]https://github.com/ArizaWilmerUTAS/System-indetification-of-underwater-vehicles-with-Multi-Output-Gaussian-Processes

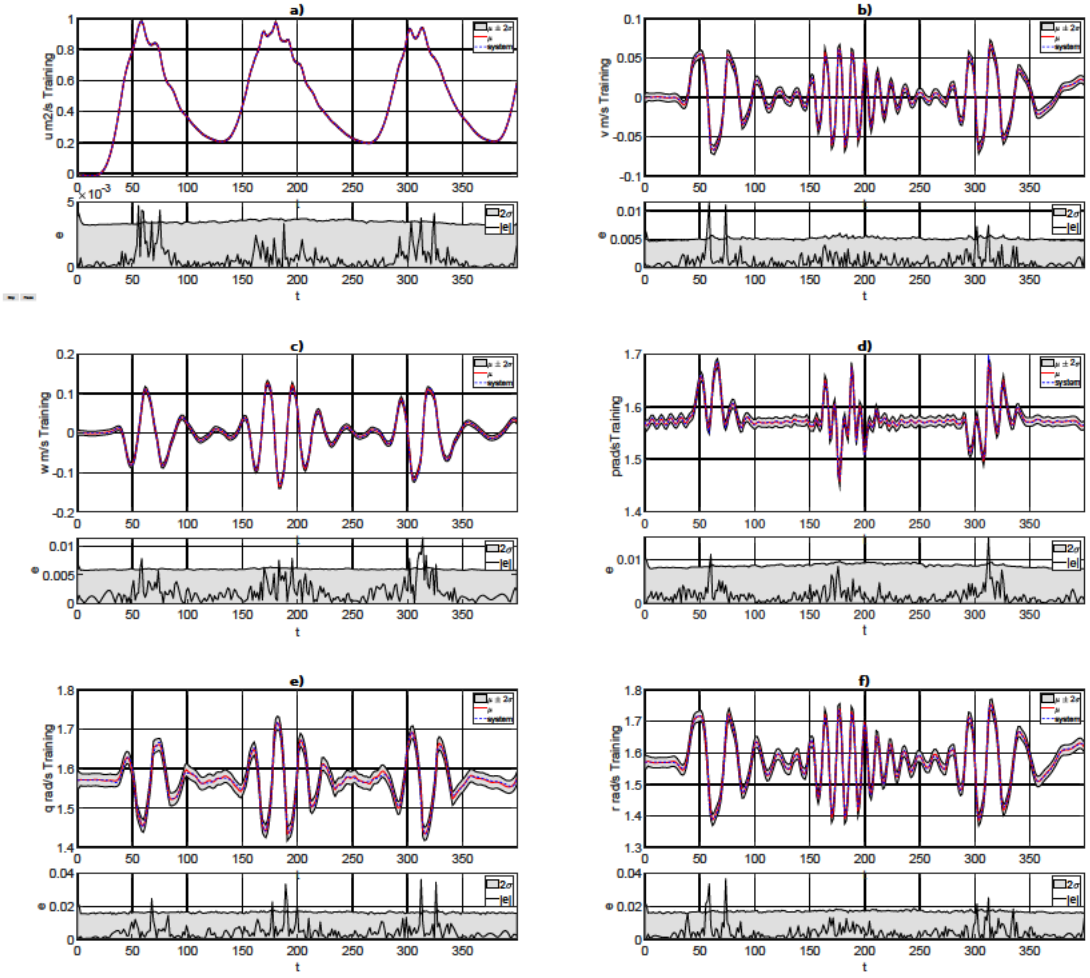

Figure 3.5: Multi-output GPs training, a) surge velocity, b) sway velocity, c) heave velocity, d) roll angular velocity, e) pitch angular velocity and f) yaw angular velocity.

expected value and encompass the error results.

The validation data consisted of the second part of the captured data in the vector form with the commanded inputs and the real output from the training data with the respective system delays. The segments of results from the validation with the second set of data are depicted in Figure 3.6. The low validation errors show a good system prediction for the sway speed and yaw speed.

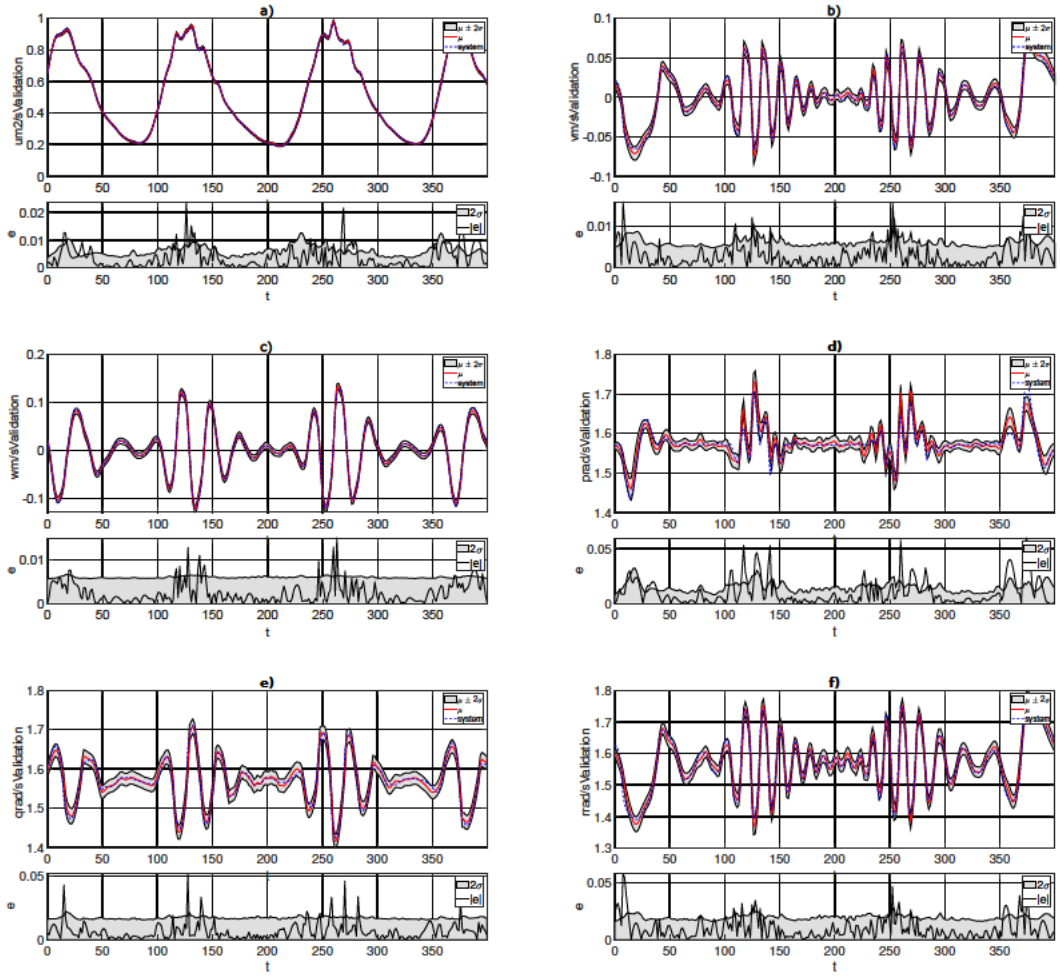

Figure 3.6: Multi-output GPs validation with unknown data, a) surge velocity, b) sway velocity, c) heave velocity, d) roll angular velocity, e) pitch angular velocity and f) yaw angular velocity.

Table 3.2: Summary prediction quality measurements.

	u	**v**	**w**	**p**	**q**	**r**	**total**
RMSE-NN	0.0507	0.0088	0.0198	0.0189	0.0242	0.021	0.037
RMSE-GP	0.0137	0.0079	0.0156	0.0161	0.0154	0.012	0.014
PRESS-NN	1.3658	0.0411	0.2077	0.1894	0.3098	0.244	2.35
PRESS-GP	0.0994	0.0334	0.1287	0.1378	0.1265	0.082	0.607

3.4.3 Simulation

With the objective to test the ability of the learning system, a simulation stage was implemented for the learned RNN and the multi-output Gaussian processes over the total length of the simulated data. Navigation applications as EKF, UKF, and control as model predictive control require predicting the behavior of the plant in a number steps ahead of the actual state to predict the correct position or control signals. In the case of the Multi-output GPs the simulation is done by feeding back to the simulation the past inputs $y_i(k-n)$, the initial position and control signals of rudder, elevator and forward speed where used, the naive simulation [12] covers training and validation data acquired from the original simulation in a close loop setup. Naive simulation provides an approximation where variances are not exactly the same, but provide a general guidance on uncertainty and this is enough for our study. If the variance is to be employed, such as in a control systems or a navigation problem, the uncertainty propagation can be included with the use a simulation based on Monte Carlo numerical approximation [12]. Figure 3.7 shows the results from the simulation of the RNN and the multi-output GPs compare to the original system. RNN and multi-Output GPs can identify the system correctly and predict the behavior of the system. However, both system identification techniques have small discrepancy in comparison to the real system; the RNN presents higher inconsistencies in the prediction of velocity in comparison to GPs that is very precise on the prediction. The better capability of GPs to predict outside the training horizon is confirmed by the results of Table 3.2. The mean value of the output root-mean-square error (RMSE) and the predicted residual error sum of squares (PRESS) for GPs are smaller than the values of a RNN.

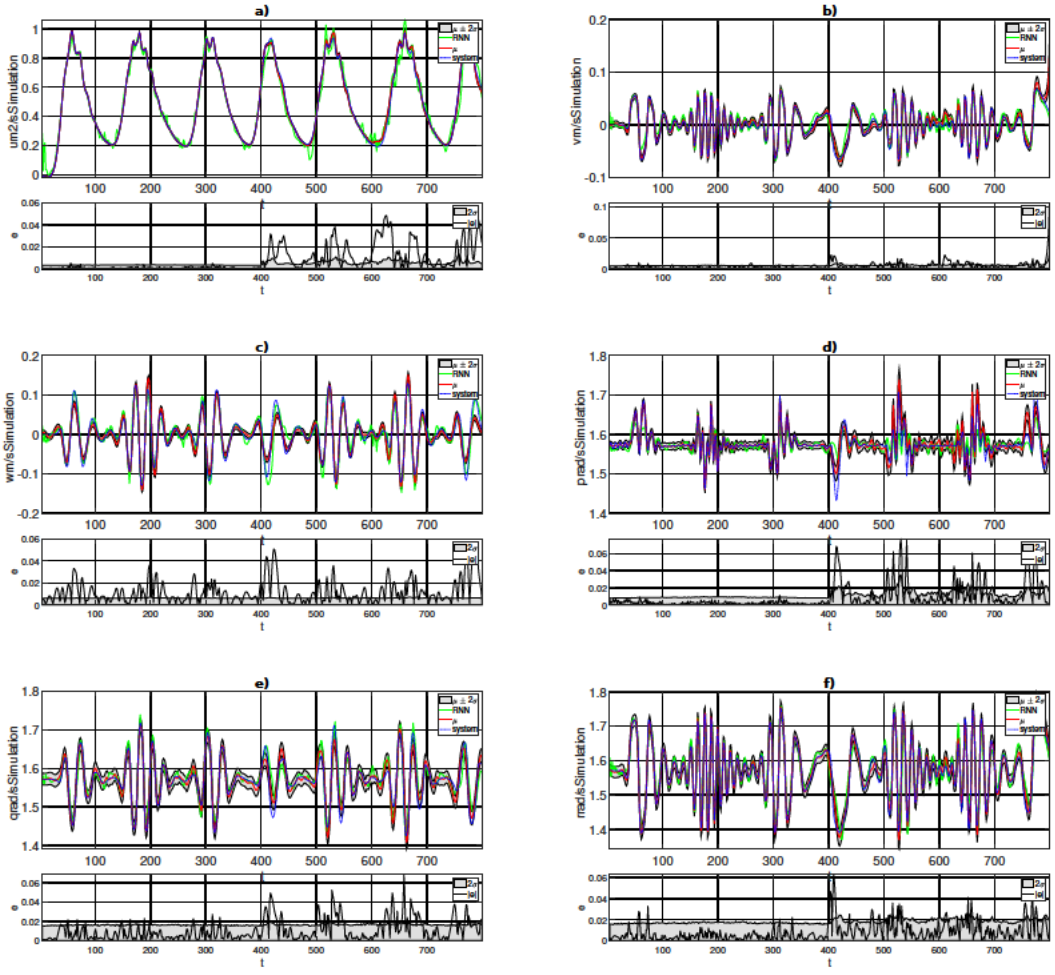

Figure 3.7: Multi-output GPs simulation compared with RNN and real system a) surge velocity, b) sway velocity, c) heave velocity, d) roll angular velocity, e) pitch angular velocity and f) yaw angular velocity.

3.5 Conclusion

In this work, the use of multi-output GPs for the system identification of AUV dynamics was tested on a REMUS 100 AUV. It was demonstrated that the nonparametric Multi-output GPs can model an AUV as well as RNN with the added value of a confidence measurement. In the simulations, GPs show a similar ability to RNN to predict and simulate the behavior of an AUV. In some cases, GPs performed better than RNN outside of the training horizons with the error between the GPs and the real system being relative low as the convolution process is equivalent to represent the system through a differential equation. The GPs model obtained also has a smaller number of hyperparameters compared to the large number of coefficients of a mathematical model.

To improve further the capability of prediction of the system, more recent suggested techniques for GPs such as Recurrent GPs can be used. The simulation of GPs can also be improved if techniques such as Montercarlo and Taylor series can take advantage of the variance to increase the horizon of cover manoeuvres and the prediction accuracy. The next work will be devoted to the development of application for navigation and control using the obtained model. As the real world is a noisy environment that can be better described with Gaussian distributions, the use of GPs can show better performance in specific tasks such as navigation and control.

This page is intentionally left blank.

Chapter 4

Machine Learning Post-Processing of Underwater Vehicle Pressure Sensor Array for Speed Measurement

The present chapter presents the novel application of dynamic non-parametric GPs model for underwater vehicles and was chosen as the platform MULLAYA was previously employed for a similar inconclusive parametric study. At the time of submission of this thesis, the present chapter is under review by the *Journal of Ocean Engineering*.

4.1 INTRODUCTION

Inertial navigation is a key element in the use of underwater vehicles (UV) as they provide information regarding the vehicle states in order to fulfil mission goals. A common inertial navigation system (INS) is composed of a set of sensors as an accelerometer, gyroscope, and compass coupled with surveillance of surrounding objects. The exact position of a vehicle is calculated by the application of sensor fusion and filters such as extended Kalman filter or unscented Kalman filter. In the case of air drones, a very accurate location can be obtained by the three sensors mentioned above for fast sampling coupled together with GPS for the drift reduction caused by the error induced by acceleration double integral. However, the accuracy and robustness of inertial navigation systems to predict the vehicle states for UVs are highly dependent upon environmental conditions. Compared to land and air drones, UVs have limited use of GPS for drift correction as the GPS signal becomes completely attenuated underwater.

In the case of UVs, INS drift correction can be undertaken by the use of specific techniques such as underwater GPS system [73] or the use of specific sensors such as Acoustic Doppler current profilers (ADCPs) and Doppler velocity logs (DVLs) [74]. ADCPs measure the current around the vehicle by the projection of 3D ultrasound wave beams and the measurement of their travelling time or frequency change, thus they can estimate the fluid velocity along the acoustic path. In a similar way, DVLs calculate the vehicle speed in relation to the sea floor, an underwater structure or object by the projection of an acoustic beam and the returning echo measurement. ADCPs and DVLs are however not fixable to small AUVs as they are large sensors and require high quantities of energy. In addition, DVLs require the presence of a surface and ADCPs do not measure the local flow on the vehicle surface thereby introducing errors in the measurements. Therefore, drift correction by using pressure sensors to measure flow speed is a promising alternative due to its small form factor in both size and energy requirements.

In nature, fish utilise a specific sense organ system, known as the lateral line system, for manoeuvring in complex fluid environments. The lateral line system is formed from a series of sensory cell clusters called the cupula which are distributed along the fish's body. These cupulas are simulated by the flow around the fish and allow it to determine the

flow rate and direction over its own body [75]. These types of microstructure were the inspiration for the application of pressure sensors in array for speed measurement and current measurement [76–78]. The relationship between the pressure and flow velocity is well known. Pitot tubes use the principle that the flow velocity can be correlated to the dynamic pressure over a surface. Based on this principal [79, 80] have developed specific micro-electromechanical system (MEMS) pressure transducers and [81] has developed a similar layout based on the hot wire principle. The goal of these developments is to convert the external vehicle surface into a multi-directional array of lateral lines.

Even though the research in pressure transducer technology has shown results, research remains limited on the precision and post-processing required for the correct calculation of UV speed from pressure sensor arrays data. It is commonly assumed in the post-processing of the pressure data that flow has a linear relationship to vehicle speed, usually in the form of parametric Bernoulli equations, i.e. zero flow acceleration. However, the work presented in this chapter shows that the flow acceleration introduces a substantial non-linearity in the relationship. This non-linearity is also hard to be characterised via regression analysis as it is both time and spatially dependent

Two machine learning methodologies were employed to improve post-processing accuracy, neural networks and Gaussian processes (GP) were tested methods to include non-linearities caused by the vehicle acceleration on the estimated speed compared to the linear parametric equation methodology. A series of CFD simulations were employed for the selection of sensors, and a towing tank experiment was carried out to explore the applicability of different post-processing methodologies. A mathematical model was developed for the integration of pressure measurements to obtain an equivalent absolute speed. The results show the need to account for the non-linearities in the post-processing of pressure data for speed measurement in underwater vehicles to obtain the correct registration of position based on these sensors.

4.2 Sensor Implementation

The MPXV5100 (Figure 4.1) 100kPa piezoresistive transducer was selected for the measurement of pressure at each point. Each sensor was coupled to a 15Ksps 20 bits analogue

to digital converter with a low-pass filter. Figure 4.2 shows the recommended configuration of a cutoff frequency of 650 Hz. The data was collected through the Xylinq FPGA of a NI myRIO 1900 at 1Ksps for each channel. The AUV was mounted directly to the AMC towing tank carriage and towed at different speeds, angles, and different acceleration rates, as shown in Table 4.1.

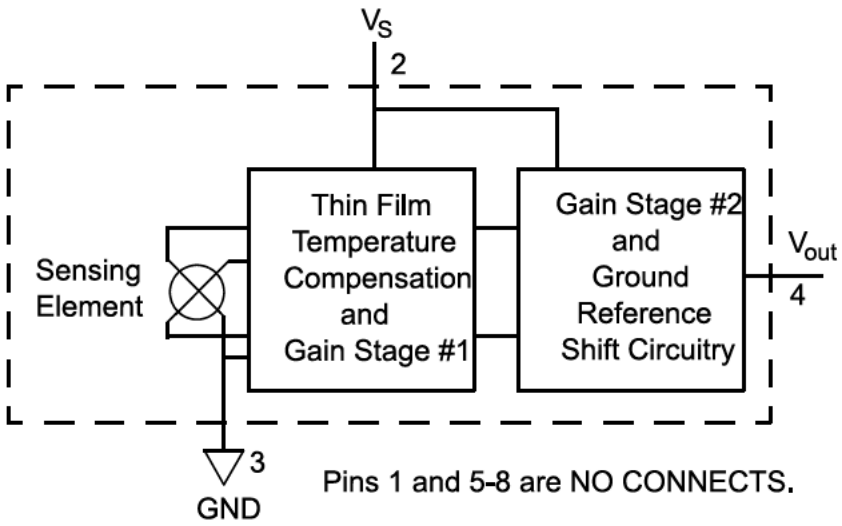

Figure 4.1: Fully integrated pressure sensor schematic [82]

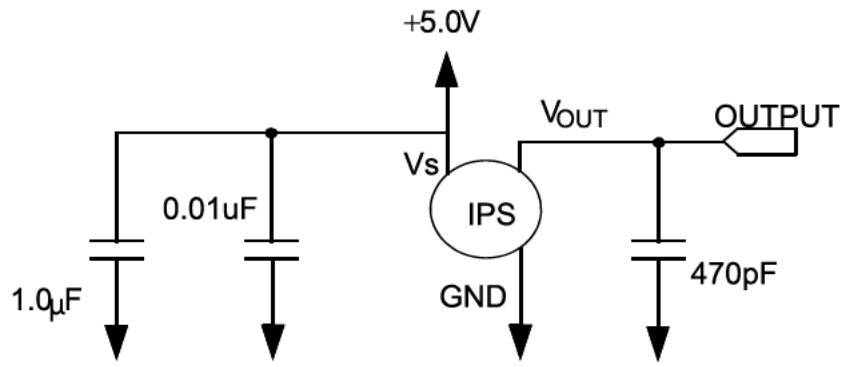

Figure 4.2: Integrated pressure sensor RC LP filter to filter out noise [83]

4.3 Linear Parametric Post-Processing

Figure 4.4 shows the five pressure sensors' locations on the AUV head. A rear sensor measures a static pressure from location P_s such that the exact robot depth can be

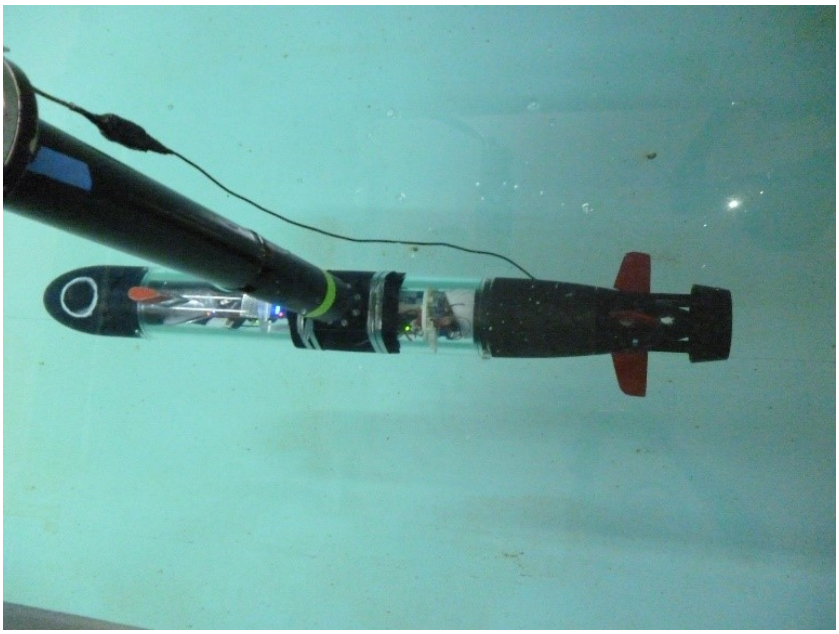

Figure 4.3: AUV at AMC towing tank

Table 4.1: Speed and angles tested in the towing tank experiment

Speed(m/s)	**Yaw(degrees)**	**Pitch(degrees)**
0	-10,-5,0,5,10	-10,-5,0,5,10
0.3	0	0
0.75	-10,-5,0,5,10	-10,-5,0,5,10
1	-10,-5,0,5,10	-10,-5,0,5,10
1.5	-10,-5,0,5,10	-10,-5,0,5,10

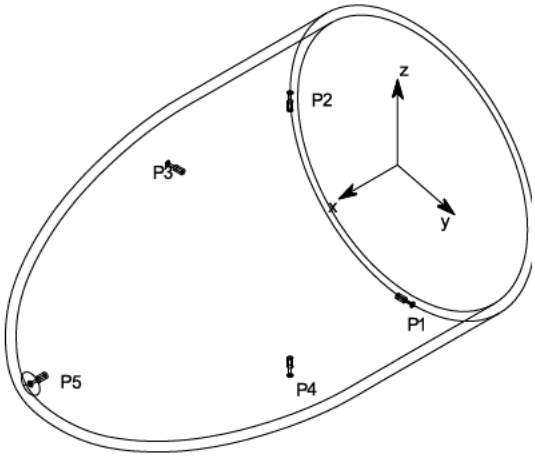

Figure 4.4: Pressure sensor distribution on the head of the AUV and the notations used

calculated as:

$$z_{P_0} = \frac{P_s}{\rho g} + \eta_2(z) * \delta_{z_{Ps-o}} \tag{4.1}$$

where, P_s is the static pressure at the sensor, $\eta_2(z)$ is the rotation matrix component from the vehicle frame to the world frame and $\delta_{z_{Ps-o}}$ is the vector of distance between the vehicle centre of gravity and the depth sensor.

As, the hydrostatic pressure for each transducer will change with the vehicle movement at each time step this pressure has to be compensated by the calculation of their respective positions in relation to the centre of gravity and the depth of the vehicle. Similarly, the depth of each transducer able to be calculated as:

$$\begin{aligned}
z_{P_1} &= \eta_2 * \mathbf{z_{P_s-P_1}} + z_{P_0} \\
z_{P_2} &= \eta_2 * \mathbf{z_{P_s-P_2}} + z_{P_0} \\
z_{P_3} &= \eta_2 * \mathbf{z_{P_s-P_3}} + z_{P_0} \\
z_{P_4} &= \eta_2 * \mathbf{z_{P_s-P_4}} + z_{P_0} \\
z_{P_5} &= \eta_2 * \mathbf{z_{P_s-P_5}} + z_{P_0}
\end{aligned} \tag{4.2}$$

where $\mathbf{z_{P_s-P_{1:5}}}$ is the vector from the depth sensor to the specific pressure sensors in the array, with the measurements P and the specific depth at each sensor, a pressure caused by the dynamic action of the vehicle P_d for each transducer can be calculated as:

$$\begin{aligned} P_{d1} &= P_1 - \rho g z_{P_1} \\ P_{d2} &= P_2 - \rho g z_{P_2} \\ P_{d3} &= P_3 - \rho g z_{P_3} \\ P_{d4} &= P_4 - \rho g z_{P_4} \\ P_{d5} &= P_5 - \rho g z_{P_5} \end{aligned} \tag{4.3}$$

The Bernoulli principle defines the relationship between the pressure and the velocity as seen in equation (4.4). The hydrostatic pressure was removed from the Bernoulli equation as this value was previously calculated. Equation (4.4) can be expanded with power functions to the form presented in equation (4.5) with the addition of matrix A of parameters defined by the AUV head shape.

$$C = \frac{\mathbf{V}^2}{2} + \frac{\mathbf{P}}{\rho} \tag{4.4}$$

$$\begin{aligned} \mathbf{V}^4 &= (C + \mathbf{A} * \mathbf{P})^2 \\ \mathbf{V} &= \sqrt[4]{(\mathbf{A}\mathbf{P}')^2 + 2(\mathbf{A}C) * \mathbf{P} + C^2} \end{aligned} \tag{4.5}$$

where

$\mathbf{A} = \begin{bmatrix} a_{(1)} & a_{(2)} & a_{(3)} & a_{(4)} & a_{(5)} \end{bmatrix}$; and C is a constant.

The parameter calculation for the mathematical model was carry out with the use of particle swarm optimisation (PSO) and minimisation of the means squared error cost function.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right) \tag{4.6}$$

where the variables used are:

m number of training examples

$x^{(i)}$ pressure values for the training prediction

$y^{(i)}$ real values of velocity

θ chosen parameters

$h_\theta \left(x^{(i)} \right)$ predicted vehicle speed.

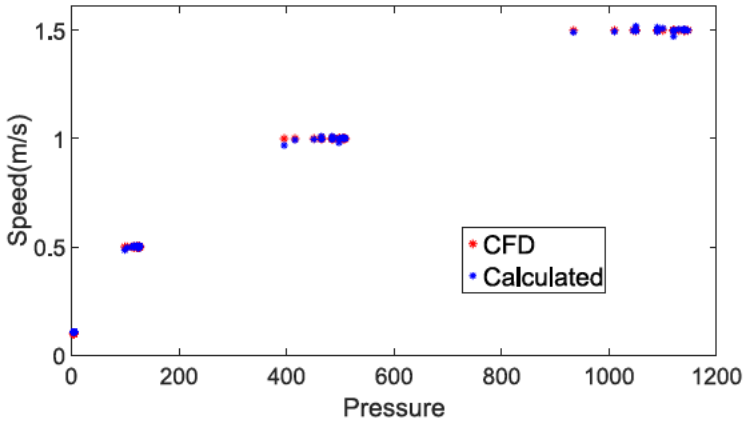

Figure 4.5: Prediction over simulated data (steady-state, linear model) for pressure sensor 5 at different speeds and angles.

The first set of CFD simulations was carried out to verify that the proposed equation can fit the steady-state data from the experiments. Figure 4.5 shows the result of speed calculation with the parameters learned over CFD data. The results of the parameter calculation over the experimental data can be seen in Figure 4.6.

4.4 Neural Network Post-Processing

Artificial neural networks (ANNs) have been successfully applied to a number of time series prediction and modelling tasks. In particular, when the time series is noisy and there is an underlying non-linear dynamical system, ANN models frequently outperform standard linear techniques. Modelling of nonlinear systems with machine learning algorithms requires the inclusion of architectures with feedback and an example of such architecture is nonlinear autoregressive models with exogenous inputs neural networks (NARX-NN) [84]. It has been shown that in concept, NARX-NN can be used instead of conventional recurrent networks, without any computational loss and that they are at least equivalent to Turing machines [85]. In general, a NARX architecture is the calculation of a function that relates the future output of the system to a series of regressors from previous states.

$$y(n) = f\left(y\left(n-1:k\right); u\left(n-1:k\right)\right) \tag{4.7}$$

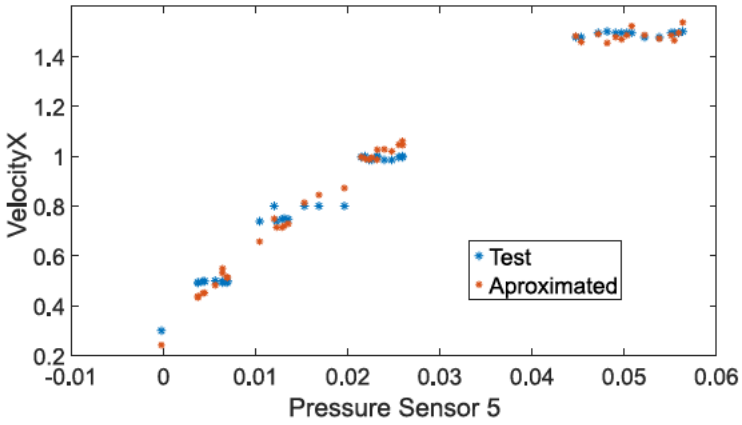

Figure 4.6: Prediction over experimental data velocity vs pressure sensor 5 (steady-state, linear model) at different speeds and angles.

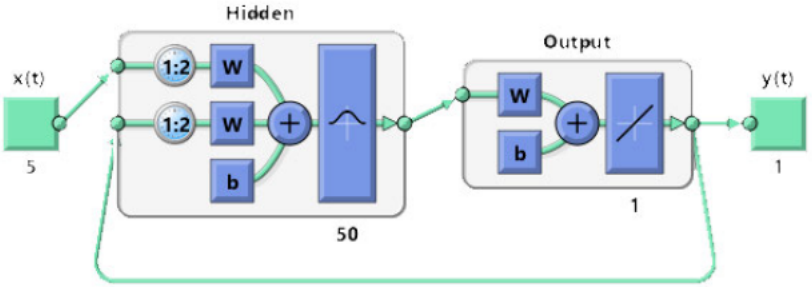

Figure 4.7: NARX-NN structure.

The nonlinear mapping $f(.)$ is generally unknown and can be estimated usually by a standard multilayer perceptron (MLP) network. There are two possibilities that have a concern in training NARX-NN. The first one is called parallel mode in which the output is feedback to the input of the feedforward neural network as part of the standard NARX architecture. The second one is called series-parallel mode in which the NARX-NN starts as an open NN in which the true output is used instead of feeding back the estimated output, after training the NARX-NN in an open loop can be transformed to a close-loop to be deployed. Parallel mode has the disadvantage of lower training times but has overall better performance in modelling a system.

For the pressure sensor data post-processing the structure illustrated in Figure 4.7 was used. The structure consists of the first and second delays of the network output and the

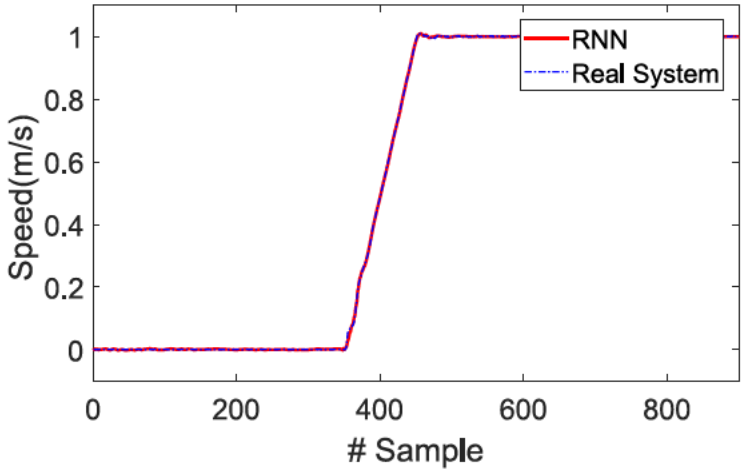

Figure 4.8: Sample training result NARX-NN, 1 m/s, 0 degrees.

pressure measurements. The training of the NARX-NN was carried out in closed-loop as it resulted in more generic training. The data obtained from the experiment was divided into training data and validation data. Experiments that were done more than one at the same speed were used for the trained model validation. The neural network training was undertaken by using Levenberg-Marquardt backpropagation of multiple series with each experiment run considered as a series and the neural network goal was to learn to predict each experiment result. A result from the training of the NARX-NN can be seen in Figure 4.8, where the NN has learned to produce the same result from the experiments by been feed with delayed signals.

4.5 GP Post-Processing

GP is another black-box methodology capable of learning the nonlinear dynamic behaviour of a system [86]. A GP is a generalisation of the Gaussian probability distribution. Where a probability distribution defines random variables, which are scalars or vectors, a stochastic process rules the properties of functions. A GP shares a relationship with NN, i.e. if an NN hidden layer is taken to infinitive, the resulting NN is an approximation to GP [87]. The principal difference of a GP compared to NNs is that a GP not only provides just the system output prediction but also a measure of prediction

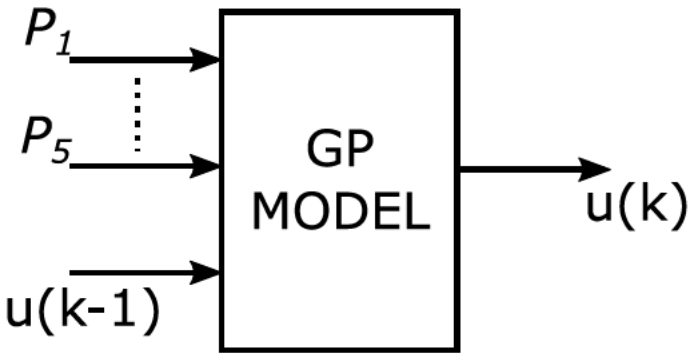

Figure 4.9: NARX structure for sensors post-processing.

confidence. A GP can be seen as an assembly of random variables $f(z_i)$ with a joint Gaussian distribution for any finite subset. A GP is a stochastic process containing random variables with a normal probability distribution. A GP is a Gaussian random function, fully described by its mean function and covariance function.

$$p(f(z_1),...,f(z_n)|\, z_1,...,z_n) = \mathcal{N}(\mathbf{m}_f, \mathbf{\Sigma}_f) \tag{4.8}$$

where m_f is the expectation of the random variables and $\mathbf{\Sigma}_f$ is the covariance of the random variables and $p(f(z_1),...,f(z_n)|\, z_1,...,z_n)$ is the joint probability function of $f(z_1),...,f(z_n)$ given $z_1,...,z_n$. The hyperparameters learning of the joint function is carried out by the minimisation of the marginal log-likelihood defined as

$$\log\ p(\mathbf{y}|\,\mathbf{X},\theta) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}| - \frac{n}{2}\log\ 2\pi \tag{4.9}$$

where $\mathbf{K} = \mathbf{\Sigma}_f + \sigma_n^2\mathbf{I}$ is the covariance matrix for the target value $\mathbf{y}$ if a white noise $v \sim \mathcal{N}(0, \sigma_n^2)$ is defined, and θ is the vector of hyperparameters [88].

The application of GP for post-processing of nonlinear systems requires the use of nonlinear model structures for black boxes as NARX (nonlinear autoregressive model with exogenous input). NARX structure (Figure 4.9) uses the input values $P_{1,...,5}(k-1)$ and the measured output values $u(k-1)$ as regressors to learn the nonlinearity behaviour. The structure NARX is simpler to deploy as the data acquired from the experiment is a series of discrete samples from the inputs and outputs and the requirement is just to apply discrete delays to the measured signals.

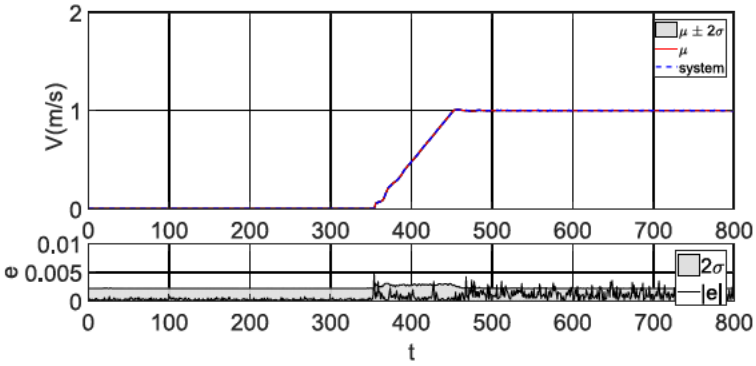

Figure 4.10: Sample training result NARX-GP.

The training of GP was carried out by the minimisation of the logarithmical marginal likelihood over the same data than NARX-NN. The minimise function of Rasmussen was employed to find the minimum value of the logarithmical marginal likelihood. The results from the training can be seen in Figure 4.10 for the condition of $1m/s$ and at 0 degrees. It is observed that the variance follows and contains the pattern of the error.

4.6 Evaluation of Models

The evaluation of the models was undertaken by simulating a real deployment application. The pressures captured in the towing tank at 0.75 m/s at 0 degrees were used to calculate the vehicle speed. All data points were provided to the trained model at each time interval. In the case of the NARX architectures, the previous calculated data point was saved for the prediction of a new vehicle speed. Section 4.6 presents the results of the three methodologies, i.e. linear methodology, NARX-NN and NARX-GP, against the experimental data. All methodologies are constrained by the absence of data at low speeds. The root-mean-square error (RMSE) (Table 4.2) summarise the results and shows that the RMSE for the NARX-GP is the smallest and that the worst results are from the linear methodology. The linear model was not trained with a low speed steady state as the noise of the towing tank and sensor resolution did not allow the capture of data. The non-linear methods were provided with data at low speed but it also important to note that the methodologies cannot be trusted at low speed (<1.0 m/s) as the noise levels affect the training. The linear methodology was found

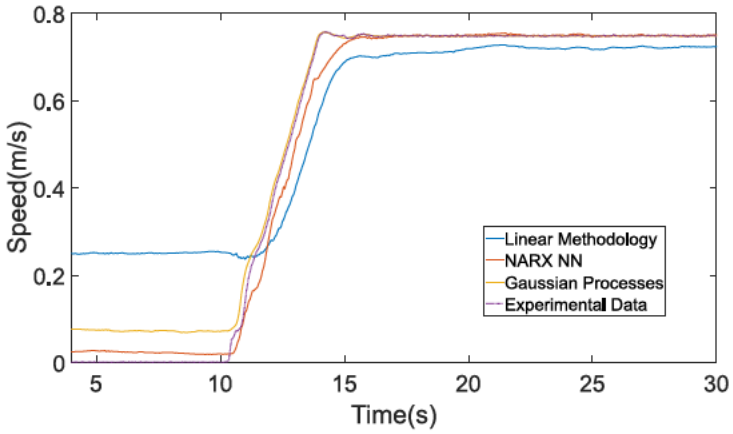

Figure 4.11: Prediction results with the linear model, NARX-NN and NARX-GP 0.75 m/s at 0 degrees.

Table 4.2: RMSE of the simulation from 13 seconds onwards

Method	**RMSE**
Linear	0.0703
NARX-NN	0.0259
NARX-GP	0.0143

to be weak in the calculating of the non-linear behaviour of the vehicle (as expected) but is also highly affected by the noise encountered in the experiment which results in a substantial overestimation for speeds less than 0.2m/s. NARX-NN shows an overall improvement in the calculation of the speed compared to the linear methodology but it does not capture the non-linear trends as precisely as the GPs model, e.g. the acceleration slope and the overshoot as the vehicle achieves the desired speed. It is also noted that the GPs methodology has another advantage by providing the standard deviation as a measurement of confidence, e.g.at low speed the variance increases representing the difficulty of detecting low speeds. Filters can exploit this measurement of confidence, such as the unscented Kalman filter.

4.7 CONCLUSION

This chapter shows that the pressure sensor array for underwater vehicles can be an important addition to the arsenal of sensors for position estimation of a UV. However, the commonly adopted linear methodology in data processing does not account for non-linear behaviour in the relationship between pressure and vehicle speed in the presence of acceleration. The experiment in the towing tank showed that acceleration has a substantial non-linear effect on the relationship. The experimental data were further used to probe the performance of non-linear methodologies to estimate vehicle speed based on pressure sensor data. The proposed non-linear methodologies, i.e. NARX-NN and NARX-GP, show the advantage of including non-linear capabilities in the post-processing of pressure measurements. NARX-NN and NARX-GP show a higher capability to predict the vehicle velocity with acceleration present. The best methodology over the experimental data was NARX-GP showing superior capability to NN in predicting the non-linear behaviour of the relationship between pressure and velocity with the addition of a confidence measurement. A future deployment of the vehicle with its own power train data is planned to test the capability of the sensors to measure the forward speed in addition to measurement of lateral speed.

Chapter 5

Navigation of Underwater Vehicles with Gaussian Processes Unscented Kalman Filter

This chapter has been published by the *Journal of the Society of Underwater Technology*[89]. The application summarise in the present chapter was chosen as an initial possible solution for the inertial navigation of MULLAYA AUV as the vehicle do not posses any algorithm for inertial navigation.

5.1 Introduction

Development of accurate and robust navigation technologies is essential for achieving high performances in underwater environments. As the need for complex missions increases, there is a growing demand for highly accurate localisation of underwater vehicles for both navigation and data collection purposes. In comparison to ground and air vehicles, localization via the Global Positioning System (GPS) is rarely available underwater. Therefore, navigation strategies which are more robust and independent from GPS are needed.

Strategies for navigation of underwater vehicles, in the simplest form, are to integrate the vehicle velocity from an accelerometer, gyroscope or water speed sensors to obtain a new position estimate [90]. If a water speed sensor is employed the position at low speeds under $0.3m/s$ cannot be established as the sensor is not capable of measuring it. In the case of inertial navigation systems, the acceleration is integrated twice with respect to time [91]. The double integral generates drift in the position result. The generated drift can be corrected using complementary sensors as Doppler Velocity Sonar (DVS), and Acoustic Doppler current profiler (ADCP) together with algorithms such as extended Kalman filter (EKF) and unscented Kalman filter (UKF).

In 1960,a Kalman Filter was introduced as an optimal solution for state estimation from a linear system by using a prediction of a physical model [92]. As most systems are non-linear, the Kalman Filter was modified to be used with non-linear system by multiple techniques generating alteration as the unscented and extended Kalman Filter (UKF, EKF). In the case of underwater vehicles, these techniques and their variations are the most popular techniques as in [93] where apart from the system states the EKF learn a calibration bias for the magnetic heading. However, applications which employ EKF have produced more robust and accurate results compared to the UKF [94–96].

Nevertheless, the good results from UKF applications for underwater vehicles, the UKF can have oddly poor performance because its predictive variances can be far too small if the sigma points are placed in unfortunate locations. Deficient predictive variance will produce observations with heavy weight in the measurement update, which causes the UKF to fit the noise [97]. Other filters were proposed for underwater vehicles as the

ensemble Kalman filter (EnKF), fuzzy Kalman Filter(FKF), and particle filters. The EnKFs represent the distribution of the system state using a random sample, called an ensemble, and replace the covariance matrix by the sample covariance computed from the ensemble [98]. A fuzzy Kalman filter is a combination of a fuzzy set with the Kalman filter, the fuzzy set is a mathematical technique to represent inaccuracies that can generate better estimation than other Kalman filter [99]. [100] compare the EnKF and FKF for underwater vehicles exhibiting better results the EnKF. The particle filter uses a different approach to the EKF by implementing Bayesian filtering. The particle filter makes an approximation of the posterior by using a finite number of particles that represent points in the solution space. Each particle is assigned a weight and the weighted sample points correspond to the solution of the posterior of the particle state. These particles are propagated according to the dynamics of the posterior and the weight is modified based on the support from the likelihood. The advantage of particle filters is that they do not require a state error Gaussian approximation. Despite, the research made to increase the particle filter speeds [101], the computational cost of running such algorithms still too high for an underwater vehicle with their internal computers.

The principal disadvantage of the mentioned filters is that their performance depend on the accuracy of the model. The calculation of coefficients from mathematical models for underwater vehicles is a complex task that requires a series of experiments [45] or CFD simulations [102] especially if such calculation or simulations are done within commercial vehicles which are modular and reconfigurable; extending the quantity of data require. Adding to the complexity of the calculation of coefficients, some coefficients are very sensitive meaning that the wrong calculation can reduce the fidelity of the predicted vehicle motion [103].

A solution to avoid the calculation of coefficients for a mathematical model is the use of non-parametric methods. [104] introduced the Gaussian processes unscented Kalman filter (GP-UKF) which is a modification of the standard UKF with the replacement of parametric models of state and measurement by non-parametric models obtained from a series of experimental test. The non-parametric models are not only capable of giving a future state prediction and measurement, but they also can give the covariance matrices Q of process and R of measurement noises. The non-parametric model learns over a

series of real experiments, and thus include more non-linearities that other common methods will avoid and it can learn to characterize the true signal over a series of noisy samples as GP has an integrated smoothing function.

This paper outlines a research into the capability of GP-UKF to predict and correct the measured states of an underwater vehicle. Equally, the require sample frequency and minimum training data proportion for the GP-UKF is presented. A Simulink model of a REMUS 100 is used to produce the training data require for the non-parametric system identification and to test the navigation algorithm. An ideal UKF and root mean square error are employed as comparison measures.

5.2 UV mathematical model

In Fossen [21] it was shown that the non-linear dynamic equations of motion of an underwater vehicle can be expressed in vector notation defined by a state vector composed of the vector v of velocities on the body frame of the form $[u, v, w, p, q, r]^T$ and the vector $[u, v, w, p, q, r]^T$ of position in the Earth fixed frame (Figure 5.1) of the form $[x, y, z, \phi, \theta, \psi]^T$ such that

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\,\mathbf{v} + \mathbf{D}(\mathbf{v})\,\mathbf{v} + \mathbf{g}(\eta) = \tau \tag{5.1}$$

with the kinematic equation

$$\dot{\eta} = \mathbf{J}(\eta)\,\mathbf{v} \tag{5.2}$$

where

η vector of position and orientation of the vehicle in Earth-fixed frame,

$\mathbf{v}$ vector of linear and angular vehicle velocities in body fixed frame,

$\dot{\mathbf{v}}$ vector of linear and angular vehicle accelerations in body fixed frame,

$\mathbf{M}$ matrix of inertial terms,

$\mathbf{C}(\mathbf{v})$ matrix of Coriolis and centripetal terms,

$\mathbf{D}(\mathbf{v})$ matrix consisting of damping or drag terms,

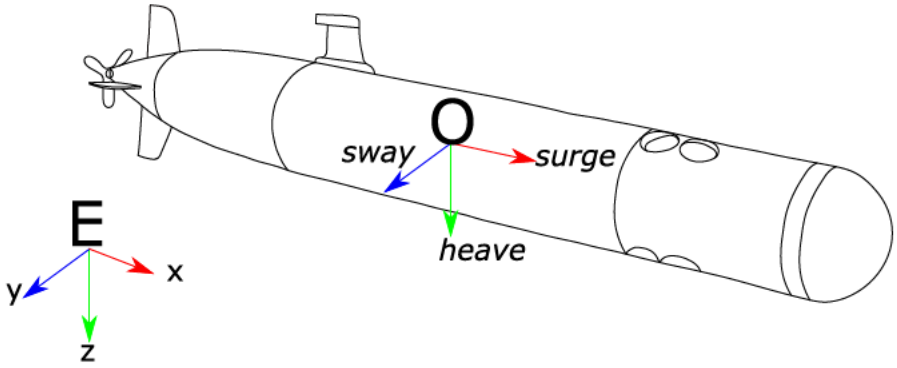

Figure 5.1: UV reference frames, vehicle frame is at center of buoyancy.

$\mathbf{g}(\eta)$ vector of restoring forces and moments due to gravity and buoyancy,

τ vector of control and external forces, and

$\mathbf{J}(\eta)$ rotation matrix that converts velocities in a body fixed frame v to an Earth fixed frame velocity $\dot{\eta}$.

Equation (5.1) can be expanded into a more general equation of motion as has been shown in [66, 105]. The expansion will be a system of six equations with 73 hydrodynamic coefficients.

5.3 UKF

Table 5.1 shows the basic structure of the UKF which estimates the states of a dynamic system based on a series of observations and an internal model. If x_k represents the state of the system, u_k represent the control input and z_k observation at time k. It can be assumed that the dynamic system evolves according to a state transition function $f(\cdot)$and an observation function $h(\cdot)$, such that

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \varepsilon_k \\ \mathbf{y}_k &= h(\mathbf{x}_k) + \delta_k \end{aligned} \tag{5.3}$$

where ε_k is additive with zero-mean Gaussian noise with covariance Q_k and, δ_k is the additive observation noise with covariance $\mathbf{R}_k$. The functions $f(\cdot)$ and $H(\cdot)$ are not linear, even when the estimate of the state $\mathbf{x}_{k-1}$ is Gaussian, the estimate after passing

Table 5.1: UKF algorithm

UKF $(\hat{x}_{k-1|k-1}, P_{k-1|k-1}, u_{k-1}, z_k, f(\cdot), h(\cdot))$

⟩Prediction

$1 : \hat{\mathbf{x}}_{k|k-1}, P_{k|k-1} \leftarrow UT\left(\hat{x}_{k-1|k-1}, P_{k-1|k-1}, u_{k-1}, f(\cdot)\right)$

$2 : P_{k|k-1} \leftarrow P_{k|k-1} + Q$

⟩Correction

$3 : \hat{z}_{k|k-1}, S_k, C_k \leftarrow UT\left(\hat{x}_{k|k-1}, P_{k|k-1}, h(\cdot)\right)$

$4 : S_k \leftarrow S_k + C_k S_k^{-1}\left(z_k - \hat{z}_{k|k-1}\right)$

$5 : \hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + C_k S_k^{-1}\left(z_k - \hat{z}_{k|k-1}\right)$

$6 : P_{k|k} \leftarrow P_{k|k-1} + C_k S_k^{-1} C_k^T$

end

the states through the transition function $f(\cdot)$ is no longer Gaussian. To estimate posteriors over the state space model, the UKF requires a stochastic approximation which is known as the unscented transform [106]. The unscented transform works by calculating a set of sigma points, whereby these points are transformed through the non-linear functions and their respective Gaussian distribution.

5.4 Regression with Gaussian Processes

Gaussian processes (GP) are a non-parametric tool from machine learning capable of learning regression functions from discrete training data. Important benefits of GPs are the ability to provide uncertainty estimates, model flexibility, and their ability to learn noise and smoothness parameters from training data [88]. A GP represents the posterior distributions over functions based on training data [107]. A GP assumes that the data is derived from a noisy process of the form

$$y_i = f(\mathbf{x}_i) + \varepsilon \tag{5.4}$$

where ε is a zero-mean additive Gaussian noise with variance σ_n^2. A test input x_* ,

conditioned in a set of data $\langle \mathbf{X}, \mathbf{y} \rangle$ will produce a Gaussian distribution with mean

$$y_* | \, \mathbf{X}, \mathbf{y}, x_* = \mathbf{K}(\mathbf{X}, \mathbf{X})[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y} \tag{5.5}$$

and variance

$$\operatorname{cov}(y_*) = k\,(x_*, x_*) - \mathbf{k}\,(x_*, X)\,[\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{k}\,(X, x_*) \tag{5.6}$$

where $k\,(x_*, x_*)$ is the evaluation of the kernel in respect to the test point x_* , $\mathbf{k}\,(x_*, X)$ is a vector defined by kernel values between x_* and the training inputs , and $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is the square kernel matrix of the training input values.

The prediction uncertainty captured by the variance depends on the process noise and the correlation between the test input and the training data. The kernel function selection is governed by application, the most widely used being the squared exponential, or Gaussian kernel which is considered a universal kernel.

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2} \right) \tag{5.7}$$

where σ^2 controls the average distance of your function away from its mean. The length scale ℓ determines the twists length in the function.

There are two principal methods for learning the hyperparameters Θ which are Bayesian model interference and marginal likelihood. Bayesian inference assumes that prior data of the unknown function to be mapped is known. A posterior distribution over the function is refined by incorporation of observations. The marginal likelihood method is based on the aspect that some hyperparameters are going to be more noticeable in their effect over the posterior distribution. Over this base the posterior distribution of hyperparameters can be described with a unimodal narrow Gaussian distribution [88].

The learning of GPs hyperparameters Θ is normally done by maximization of the marginal likelihood. The marginal likelihood can be expressed as:

$$p\,(\mathbf{y}\,|\mathbf{x}, \boldsymbol{\Theta}\,) = \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{K}|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathbf{y}^T \mathbf{K}^{-1} \mathbf{y}} \tag{5.8}$$

where N is the number of input learning data points and y is a vector of learning output data of the form $[y_1; y_2; \cdots y_N]$. To reduce the calculation complexity, it is

preferred to use the logarithmical marginal likelihood that is obtained by the application of logarithmic properties to (Equation (5.8)).

$$\mathcal{L}(\mathbf{\Theta}) = -\frac{1}{2}\log(|\mathbf{K}|) - \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{N}{2}\log(2\pi) \tag{5.9}$$

To find a solution for the maximization of log-likelihood multiples methods of optimization can be applied, e.g. particle swarm optimization, genetic algorithms, or gradient descent. For deterministic optimization methods, the computation of likelihood partial derivatives with respect to each hyperparameter is needed. From [71] log-likelihood derivatives for each hyperparameter can be calculated by:

$$\frac{\partial\mathcal{L}(\mathbf{\Theta})}{\partial\mathbf{\Theta}_i} = -\frac{1}{2}trace\left(\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\right) + \frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\frac{\partial\mathbf{K}}{\partial\mathbf{\Theta}_i}\mathbf{K}^{-1} \tag{5.10}$$

5.5 GP-UKF

The objective of the GP-UKF is to replace the internal parametric model f used for state calculation and observation model h with a non-parametric model generated by GPs and the use of the respective variance for the calculation of $\mathbf{Q}_k$ and $\mathbf{R}_k$. The process noise covariance is obtained from the predictive GP uncertainty at the previous mean sigma point, and is used for the calculation of the sigma points. This sigma points are passed through the GP observation model. The observation error covariance is obtained from the observation GP.

Table 5.2 shows the basic structure of the GP-UKF algorithm. The incorporation of GP regression allows GP-UKF to learn their models and noise processes from training data. Additionally, the noise models of the filter automatically adapt to the system states, depending on the density of training data around the current state. Consequently, if the calculation is outside of the identified region, the GP-UKF produces higher uncertainty estimates, reflecting the higher uncertainty in the underlying process model.

Table 5.2: GP-UKF algorithm

$\mathrm{GP-UKF}\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{z}_k, \mathrm{GP-f}\left(\cdot\right), \mathrm{GP-h}\left(\cdot\right)\right)$

⟩Prediction

$1: \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \mathbf{Q}_k \leftarrow UT\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathrm{GP-f}\left(\cdot\right)\right)$

$2: \mathbf{P}_{k|k-1} \leftarrow \mathbf{P}_{k|k-1} + \mathbf{Q}_k$

⟩Correction

$3: \hat{\mathbf{z}}_{k|k-1}, \mathbf{S}_k, \mathbf{C}_k, \mathbf{R}_k \leftarrow UT\left(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \mathrm{GP-h}\left(\cdot\right)\right)$

$4: \mathbf{S}_k \leftarrow \mathbf{S}_k + \mathbf{R}_k$

$5: \hat{\mathbf{x}}_{k|k} \leftarrow \hat{\mathbf{x}}_{k|k-1} + \mathbf{C}_k\mathbf{S}_k^{-1}\left(\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}\right)$

$6: \mathbf{P}_{k|k} \leftarrow \mathbf{P}_{k|k-1} + \mathbf{C}_k\mathbf{S}_k^{-1}\mathbf{C}_k^T$

end

5.6 Test Setup and Results

A simulation model of a REMUS 100, (Figure 5.2) based on the work of [66] and [72], was developed in the MATLAB/Simulink software environment and employed to produce data for test and training of the GP [72].Block diagram of the REMUS 100 model is shown in Figure 5.2. A path following controller [108] composed of a line-of-sight(LOS) law that pursue a point $P(t)$ and three robust PID controllers produce the require signals of RPM, elevator force and rudder force require to control the vehicle. The controllers employ the corrupted measurement to calculate the require forces. A sample frequency of 5Hz was used to capture data, a sub sample of 40% of the data was taken randomly for the training. The training data has more points at the start of the trajectory and reduce the quantity of points over time. Figure 5.3 shows the selected data for training compare to the simulation data. The virtual sensors employed were a 3 axis gyroscope, a 3 axis accelerometer, a compass and a DVL unit; the measurement results is the vector $[u, v, w, p, q, r, Z, \theta, \psi, \phi]$. Each sensor was simulated by a model composed of an additive noise source and the digitalization of the measurement through a 12 bits ADC. A helix movement was employed to capture 800 seconds. The AUV is accelerated from a initial velocity of $0.5m/s$ to $1.4m/s$. Figure 5.4 shows the recorded command signals for 800

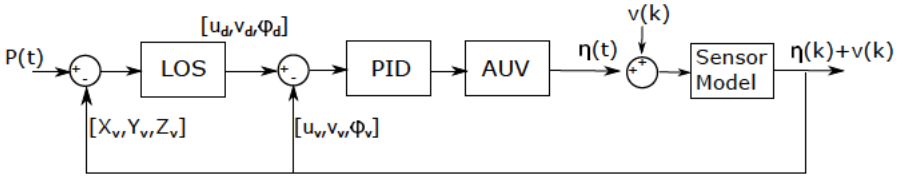

Figure 5.2: REMUS 100 simulation model, X_v, Y_v, Z_v are the vehicle position

seconds. The noise in the depth sensor required a hard response by the integral parts of the PID such that the vehicle converges to the desire path. A total of 20 simulations were carry out to capture data. The first set of data was employed for the creation of the non-parametric GP model.

The state vector was defined as the combination of vehicle speeds and vehicle position $\eta = [u, v, w, p, q, r, X, Y, Z, \theta, \psi, \phi]$. An unscented Kalman filter was also implemented as an evaluation measure, the filter uses the original REMUS 100 model from which the data was captured, this allows to compare the GP-UKF to an ideal UKF where all parameters from the vehicle are known. The GP-UKF and the UKF will have to estimate the x and y states from the vehicle. The algorithms of [109] were employed with the small modification to the GP such that a structure NARX(Nonlinear autoregressive network with exogenous inputs) can be utilize for SI(system identification) of the system. The modification consisted in the assemble of the input vector for learning as $Xd = [\eta, ui]$, where η is the state vector and ui is the command signal; and the output vector is formed of the delay vector $Yd = Xd(k-1, k-2)$.

Figure 5.5 shows the comparison between the measured data of the real vehicle state, the UKF, and the GP-UKF. The GP-UKF is equally capable of correct the measurement state of the vehicle as the positioning of the sigma points is estimated from the GPs dynamic model. In the same way the predicted position states (Figures 5.6 and 5.7) from the GP-UKF x and y have a higher similarity to the vehicle real position as the GPs employ data corrupted by noise and it has learned to predict over this data.

The comparison of the vehicle position estimations and real position is shown in Figure 5.8. The UKF shows a drift in the calculation of x and y over time. In comparison to the UKF, the GP-UKF has clearly better performance in the prediction of the vehicle position in terms of both the horizon of the training data from the GP and decay outside

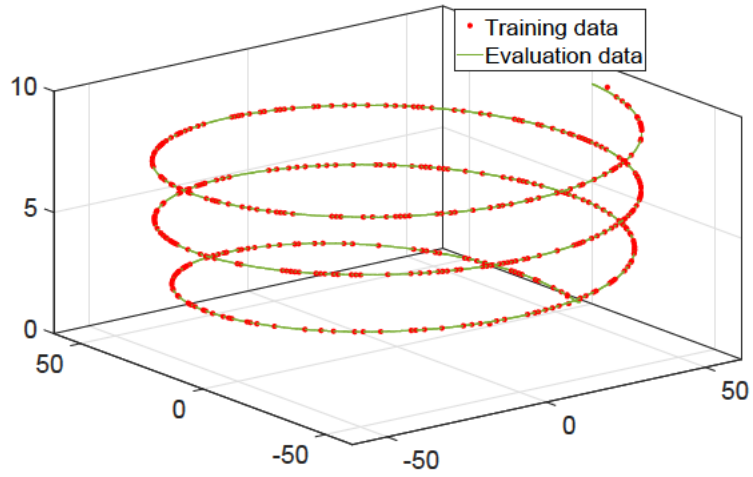

Figure 5.3: Training and evaluation data

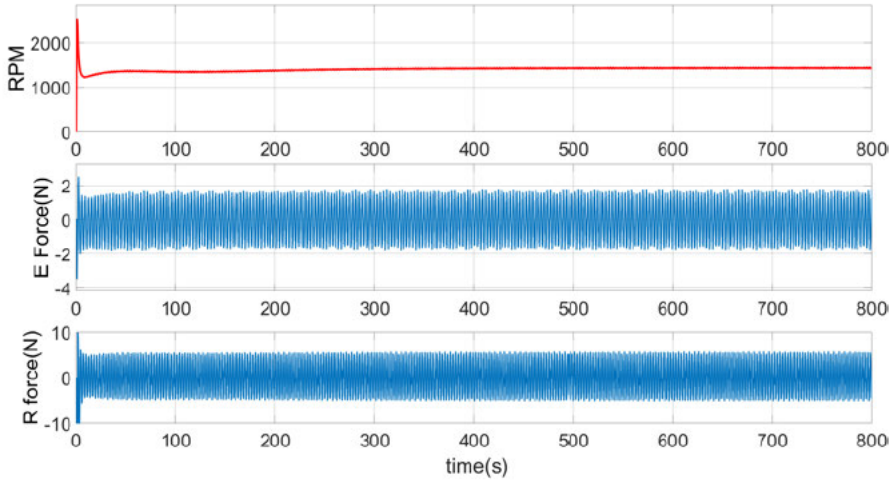

Figure 5.4: Helix test input signals to control surfaces.

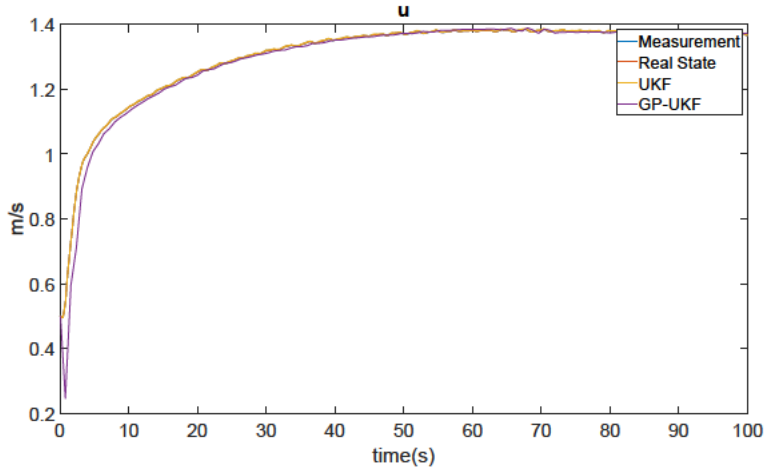

Figure 5.5: Surge speed comparison between corrupted measure data, real position, UKF and GP-UKF.

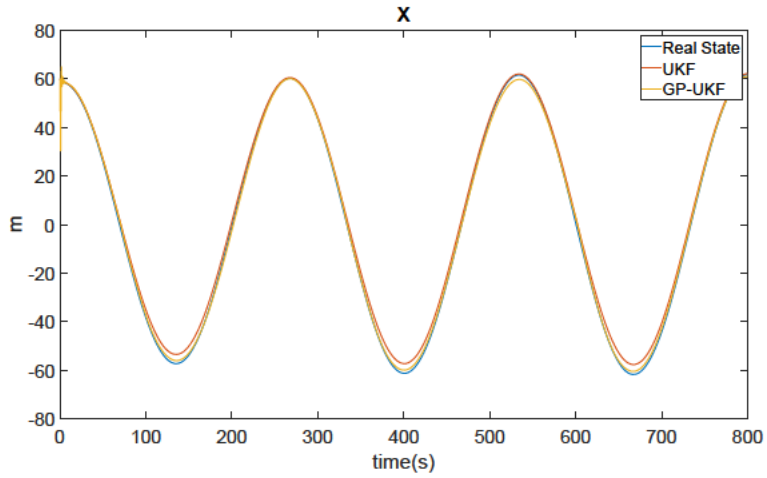

Figure 5.6: Predicted position state x comparison between UKF and the GP-UKF.

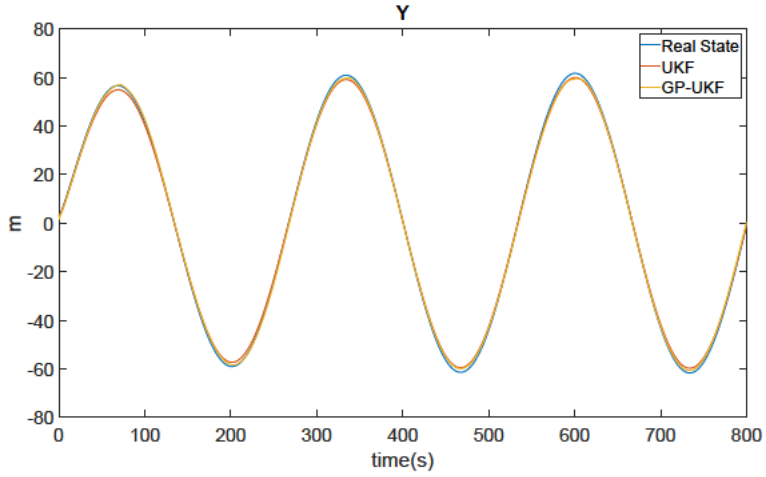

Figure 5.7: Predicted position state y comparison between UKF and the GP-UKF.

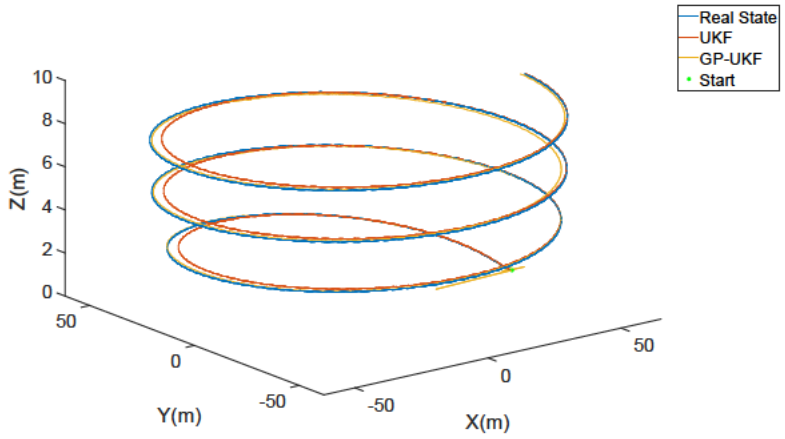

Figure 5.8: Comparison of position estimation between real state, UKF and GP-UKF.

of the training horizon. Although the error increases outside of the training horizon, this error is corrected by the filter in the return to the training horizon.

Table 5.3 and Table 5.4 summarize the measurement of the mean root-mean-square error (RMSE) between the real vehicle states and the correction from the UKF and the GP-UKF for the 20 runs. The results confirm that GP-UKF can perform as reliably as an ideal UKF and in some cases, can over-perform the ideal UKF. Equally, Table 5.4 shows the average measurement for correction and prediction. In the case of prediction the GP-UKF can forecast the position of vehicle with higher precision than an ideal UKF, the reason to this is that the inclusion of the noise function and the smooth prediction

Table 5.3: Mean RMSE and standard deviation results from UKF and GP-UKF per state for 20 runs

	$u(m/s)$	$v(m/s)$	$w(m/s)$	$p(rad/s)$	$q(rad/s)$	$r(rad/s)$
UKF	0.0014	0.0015	0.0005	0.0023	0.0013	0.0031
σ-UKF	0.0003	0.0021	0.0008	0.0078	0.0035	0.0121
GP-UKF	0.0158	0.0369	0.0154	0.0569	0.0601	0.1813
σ-GP-UKF	0.0387	0.0069	0.0033	0.0123	0.0130	0.0347
	$X(m)$	$Y(m)$	$Z(m)$	$\phi(rad)$	$\theta(rad)$	$\psi(rad)$
UKF	2.4237	1.2695	0.0017	0.0006	0.0012	0.0013
σ-UKF	14.6908	8.5834	0.0019	0.0020	0.0016	0.0988
GP-UKF	2.0541	1.3894	0.0281	0.0235	0.0297	0.1095
σ-GP-UKF	11.1893	9.9445	0.1889	0.0047	0.0052	0.3848

Table 5.4: Mean RMSE results from UKF and GP-UKF for correction and prediction

RMSE	**Value**
RMSE-UKF correction	0.0028
RMSE-GP-UKF correction	0.086
RMSE-UKF prediction	2.91
RMSE-GP-UKF prediction	2.86

of the GPs benefict the ability of the UKF to predict position .

5.7 Conclusion

Through this research, it has been seen that the GP-UKF is a promising approach for state estimation in applications where accurate parametric model is not available. The GP-UKF shows similar performance compared to an ideal UKF in the prediction and correction of the vehicle states for the helix movement test case. The average RMSE(Table 5.4) for prediction and correction demonstrate that non-parametric models can be employ as prediction model inside of Kalman filter as the unscented Kalman filter for AUVs.

GP-UKF show a better performance in the prediction of states than UKF. The smoothing kernel of the GP insured a smooth transition between the prediction points and a better placement of sigma points. The tuning complexity in the implementation of a non-linear Kalman filter are reduced dramatically as there is not require for the user to produce the covariance matrices for the process and measure noise model. The GP-UKF can be converted to an important tool for underwater vehicles, especially in cases where high non-linearities are expected as in operation near surface, near another object or specialized missions. Another advantage of GP-UKF is that it can be used even when no GPS signal is available, , which is essential for correction in traditional Kalman filters. An underwater vehicle can switch between filters as the availability of data is reduced.

The principal disadvantage of GP models is their computational cost during training. Nonetheless, research has shown that this cost can be reduced using sparse GP models allowing the use of more complex models or configurations. Work is currently ongoing to prepare an autonomous underwater vehicle for a series of experiments to train its GP-based non-parametric model. Once trained and verified, the vehicle will be deployed to assess the performance of the GP-UKF outside a controlled environment.

This page is intentionally left blank.

Chapter 6

Semi-Empirical Calculation of Hydrodynamic Coefficient for AUV MULLAYA

This chapter calculates the mathematical model of MULLAYA AUV to be employed in the simulation for model based RL of chapter 7.

6.1 Introduction

This chapter describes the calculation of a mathematical model for MULLAYA underwater vehicle in six degrees of freedom, its comparison with previous similar vehicles, and a series of experiments that were conducted. In this model, the external forces and moments resulting from hydrostatics, hydrodynamic lift and drag, added mass, and the control inputs of the vehicle propeller and fins are all defined in terms of vehicle coefficients. The calculation of the coefficients is conducted by replicating the work done [66] by creating a script capable of calculating the coefficients. This model will be employed in Chapter 7 for the learning of policies for motion control of MULLAYA AUV. The calculation of the coefficients for an underwater vehicle can be undertaken in several ways. Experimental techniques such as planar motion mechanics isolate particular characteristics to allow the calculation of coefficients, and vehicle trials paired with observers such as the Kalman filter allow the calculation of the coefficients. Other modern common techniques are the use of CFD. Although CFD methodology has gained popularity, there is a dependency on the correct user input [110]. Other methods that are commonly used are the semi-empirical methodologies which use experimentally derived guidelines for estimating model parameter values for vehicles with generic shapes. Analytical methods for calculation of model parameter such as strip theory or solving Laplace's equation can be implemented. Strip theory, also known as slender body approximation, can estimate the hydrodynamic coefficients for a body using 2D sectional properties. Strip theory can also approximate other coefficients in the equations of motion, such as damping coefficients [111].

In this chapter a combination of semi-empirical techniques will be used for the calculation of the coefficients, and the verification of the methodology is carried out by replicating calculations from previous work and by simulating the vehicle behaviour and comparing it to a series of experiments. The equations determining the coefficients, as well as those describing the vehicle rigid-body dynamics, are left in non-linear form to better simulate the inherently non-linear behaviour of the vehicle.

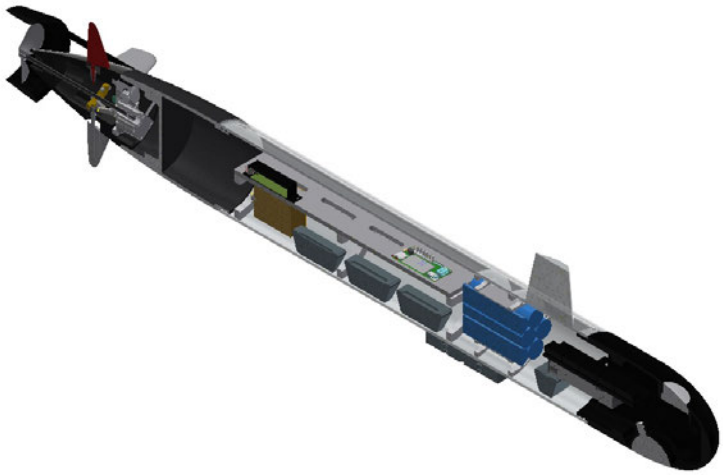

Figure 6.1: MULLAYA 3D CAD model.

6.2 Vehicle Properties

MULLAYA AUV is a vehicle designed by the Defence Science and Technology Group(DST Group) as a custom vehicle for technology testing [112, 113]. The vehicle, which is in a process of reconstruction and systems update, was lent to the Australian Maritime College as a tool for research. The 3D CAD model of the vehicle (Figure 6.1) was updated with the new weight distribution of the changes made to the platform which includes the location of the lead weights. This updated model allows the calculation of the weight and moment of inertia around the centre of gravity. The buoyancy is calculated by the construction of a solid 3D model equivalent that can also be employed for CFD simulations. The results from this initial calculation of the vehicle can be seen Table 6.1. The characteristics of the control surfaces are displayed in Table 6.2.

6.3 Vehicle Shape Generation

The application of slender body theory requires the creation of a 2D dimensional profile of the vehicle. In this case, an STL file was exported to Matlab and the technique of [114] was applied to create a 2D profile (Figure 6.2). This profile is translated and

Table 6.1: Vehicle properties of MULLAYA

Property	**Value**	**Unit**
Vehicle Length	1.56	m
Diameter	0	m
Buoyancy	246.2	N
Weight	239.364	N
Start of fins	0	m
End of fins	0	m
Ixx	0.084	kg. m^2
Iyy	3.084	kg. m^2
Izz	3.062	kg. m^2

Table 6.2: Fin properties of MULLAYA

Property	**Value**	**Unit**
S_{fin}	0	m^2
b_{fin}	0	m
$x_{finpost}$	0	m
$\delta_{\max}$	11	deg
a_{fin}	0	m

rotated such that the centre of gravity is located at coordinates $< 0, 0 >$. The upper half of the profile will be used as the function $R(x)$ as content the radius at each sliced section. This 2D profile is required to be re-sample in the vehicle's length at least 5000 to obtain the convergence of the coefficients [110] (Figure 6.3).

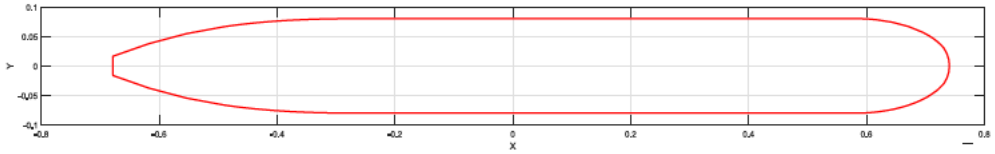

Figure 6.2: MULLAYA exported 2D profile

6.4 Added Mass

One aspect of modelling AUVs is the calculation of the added mass. The coefficient of added mass is the representation of the additional required force needed to move a vehicle in a fluid compared to moving it in a vacuum. A vehicle moving in a fluid has to push the particles of the fluid and accelerate them away from the vehicle. This need to push particles generates an increase in inertia that is related to the vehicle shape.

$$\begin{aligned} F_j &= -\dot{u}_i m_{ji} - \varepsilon_{jkl} u_i \Omega_k m_{li} \\ Mj &= -\dot{u}_i m_{j+3,i} - \varepsilon_{jkl} u_i \Omega_k m_{l+.3,i} - \varepsilon_{jkl} u_k u_i m_{li} \\ where\ & i = 1, 2, 3, 4, 5, 6 \\ and\ & jkl = 1, 2, 3 \end{aligned} \tag{6.1}$$

The alternating tensor ε_{jkl} is equal to $+1$ if the indices are in cyclic order $(123, 231, 312)$,-1 if the indices are acyclic $(132, 213, 321)$, and zero if any pair of the indices are equal [21, 115]. As the vehicle has a top-bottom and port-starboard symmetry, the vehicle

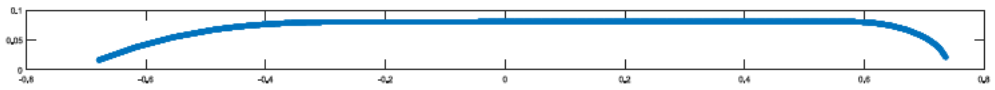

Figure 6.3: MULLAYA processed 2D profile

added mass matrix can be reduced to:

$$\begin{bmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & N_{\dot{v}} \\ 0 & 0 & Z_{\dot{w}} & 0 & M_{\dot{w}} & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & Z_{\dot{q}} & 0 & M_{\dot{q}} & 0 \\ 0 & Y_{\dot{r}} & 0 & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{6.2}$$

6.4.1 Axial Added Mass

The approximation of the axial added mass can be undertaken by the approximation of the hull shape to an ellipsoid for which the major axis is half the vehicle length l, and the minor axis is the vehicle radius $d/2$. [116] cited on [66] gives the empirical formula for the axial added mass of an ellipsoid.

$$X_{\dot{u}} = -\frac{4\beta\rho\pi}{3}\left(\frac{d}{2}\right)^3 \tag{6.3}$$

where ρ is the density of the fluid, β is an empirical value measure by [116] that is dependent on the ratio between the vehicle length and the diameter.

6.4.2 Crossflow Added Mass

MULLAYA AUV's added mass was calculated using strip theory on a mixture of cylindrical and cruciform hull cross sections. [115] established that the added mass per unit of length of a cylinder is given as:

$$m_a(x) = \pi\rho R(x)^2 \tag{6.4}$$

where ρ is the density of the fluid, and $R(x)$ is the vehicle side profile as calculated in section 6.3. The added mas of a circle with fins is given as follows:

$$m_{af}(x) = \pi\rho\left(a_{fin}^2 - R(x)^2 + \frac{R(x)^4}{a_{fin}^2}\right) \tag{6.5}$$

where a_{fin}, as defined in Table 6.2, is the maximum height above the vehicle of the vehicle fins. If equation (6.4) and equation (6.5) are integrated over the vehicle length

Table 6.3: Integration limits for MULLAYA (m)

Property	**Value**
x_t	0
x_f	0
x_f2	0
$xb2$	0

with the limits of Table 6.3, the following equations can then be employed for the calculation of each cross flow coefficient:

$$
\begin{aligned}
Y_{\dot{v}} &= -\int_{x_t}^{x_f} m_a(x)\,dx - \int_{x_f}^{x_{f2}} m_{af}(x)\,dx - \int_{x_{f2}}^{x_{b2}} m_a(x)\,dx \\
Z_{\dot{w}} &= Y_{\dot{v}} \\
M_{\dot{w}} &= -\int_{x_t}^{x_f} x m_a(x)\,dx - \int_{x_f}^{x_{f2}} x m_{af}(x)\,dx - \int_{x_{f2}}^{x_{b2}} x m_a(x)\,dx \\
N_{\dot{v}} &= -M_{\dot{w}} \\
Y_{\dot{r}} &= N_{\dot{v}} \\
Z_{\dot{q}} &= M_{\dot{w}} \\
M_{\dot{q}} &= -\int_{x_t}^{x_f} x^2 m_a(x)\,dx - \int_{x_f}^{x_{f2}} x^2 m_{af}(x)\,dx - \int_{x_{f2}}^{x_{b2}} x^2 m_a(x)\,dx \\
N_{\dot{r}} &= M_{\dot{q}}
\end{aligned}
\tag{6.6}
$$

6.4.3 Rolling Added Mass

The estimation of the rolling added mass is conducted by assuming that the vehicle hull and other surfaces except fins do not generate added mass in roll motion. [116] formulated that the added mass of a rolling circle with fins is:

$$
K_{\dot{p}} = -\int_{x_f}^{x_{f2}} \frac{2}{\pi}\rho a^4 dx \tag{6.7}
$$

where a is the fin height above the vehicle centreline.

6.4.4 Body Lift

Vehicle body lift is the result of the vehicle moving through the fluid at an angle of attack, causing a drop in pressure along the stern, upper section of the vehicle hull. This pressure drop is modelled as a point force applied. As this force does not line up with the vehicle centre of buoyancy, this force also leads to a pitching moment about the origin. [66] established that [117] is the most accurate methodology for lift coefficient calculation. The lift coefficient can be calculated as:

$$Y_{uvl} = Z_{uwl} = -\frac{1}{2}\rho d^2 c_{yd\beta} \tag{6.8}$$

where $c_{yd\beta}$ is calculated as:

$$c_{yd\beta} = c^{\circ}_{yd\beta}\left(\frac{180}{\pi}\right) \tag{6.9}$$

and $c^{\circ}_{yd\beta}$ is approximated as:

$$for\ 6.7 \leq \frac{l}{d} \leq 10,\ c^{\circ}_{yd\beta} = 0.003 \tag{6.10}$$

in the same way the moment of the force can be calculated as :

$$M_{uwl} = -\frac{1}{2}\rho d^2 c_{yd\beta} x_{cp} \tag{6.11}$$

where x_{cp} is at a point between 0.6 and 0.7 of the total body length from the nose.

6.4.5 Fin Lift

The attitude of MULLAYA AUV is similar to REMUS vehicle as it is controlled by two stern planes and two rudders. The pairs of fins move together; stern planes do not move independently of each other, nor do the rudder planes. For the vehicle control fins, the empirical formula for fin lift forces is given as:

$$\begin{aligned} Y_{uu\delta r} &= -Y_{uvf} = \rho c_{L\alpha} S_{fin} \\ Y_{uu\delta s} &= Z_{uwf} = -\rho c_{L\alpha} S_{fin} \\ Y_{urf} &= -Z_{uqf} = -\rho c_{L\alpha} S_{fin} x_{fin} \end{aligned} \tag{6.12}$$

and moments as

$$\begin{aligned} M_{uu\delta s} &= M_{uwf} = \rho c_{L\alpha} S_{fin} x_{fin} \\ N_{uu\delta r} &= -N_{uwf} = \rho c_{L\alpha} S_{fin} x_{fin} \\ M_{uqf} &= N_{urf} = -\rho c_{L\alpha} S_{fin} x^2_{fin} \end{aligned} \tag{6.13}$$

where $c_{L\alpha}$ is the fin coefficient calculated as in [117], S_{fin} is the fin platform area, and x_{fin} is the coordinate to the centre of the fin.

6.5 Hydrodynamic Damping

Hydrodynamic damping refers to the viscous contribution to damping arising from flow around the hull. The principal components of hydrodynamic damping is produced by skin friction due to boundary layers and damping due to vortex shedding.

6.5.1 Axial Drag

The vehicle non-linear axial drag coefficient can be expressed as [66]:

$$X_{u|u|} = -\frac{1}{2}pc_dA_f \tag{6.14}$$

where ρ is the density of the surrounding fluid, A_f is the vehicle frontal area, and C_d the axial drag coefficient of the vehicle. C_d can be approximated as[118]:

$$c_d = \frac{c_{ss}\pi A_p}{A_f}\left[1 + 60\left(\frac{d}{l}\right)^3 + 0.0025\left(\frac{l}{d}\right)\right] \tag{6.15}$$

where $A_p = ld$ is the vehicle plan area, C_ss is Schoenherr's value for a flat plate skin friction that was approximated in the same way as in [66], and A_f is the vehicle frontal area.

6.5.2 Crossflow Drag

The vehicle crossflow drag coefficients are the sum of the hull crossflow drag plus the fin crossflow drag. The method used for calculating the hull drag is analogous to strip theory[66]. Slender body theory is considered to be an accurate method for calculating added mass, but for viscous terms it can be imprecise[118]. However, an approximation is required to generate a first iteration of the mathematical model to allow future correction

of the values. The non-linear drag coefficients can be calculated as:

$$\begin{aligned}
Y_{v|v|} &= Z_{w|w|} = -\tfrac{1}{2}\rho c_{dc} \int_{x_t}^{x_{b2}} 2R\,(x)dx - 2 \cdot \left(\tfrac{1}{2}\rho S_{fin} c_{df}\right) \\
M_{w|w|} &= -N_{v|v|} = \tfrac{1}{2}\rho c_{dc} \int_{x_t}^{x_{b2}} 2xR\,(x)dx - 2x_{fin} \cdot \left(\tfrac{1}{2}\rho S_{fin} c_{df}\right) \\
Y_{r|r|} &= -Z_{q|q|} = -\tfrac{1}{2}\rho c_{dc} \int_{x_t}^{x_{b2}} 2x\,|x|\,R\,(x)dx - 2x_{fin}\,|x_{fin}| \cdot \left(\tfrac{1}{2}\rho S_{fin} c_{df}\right) \\
M_{q|q|} &= N_{r|r|} = -\tfrac{1}{2}\rho c_{dc} \int_{x_t}^{x_{b2}} 2x^3 R\,(x)dx - 2x_{fin}^3 \cdot \left(\tfrac{1}{2}\rho S_{fin} c_{df}\right)
\end{aligned} \tag{6.16}$$

where ρ is the fluid density, C_de the drag coefficient of a cylinder, and C_df the crossflow drag coefficient of the control fins.

6.5.3 Rolling Drag

In the same way that the calculated rolling added mass is calculated, the rolling added drag coefficient can be calculated as the rolling drag of the fins.

$$K_{p|p|} = Y_{vvf} r_{mean}^3 \tag{6.17}$$

where Y_{vvf} is the fin component of the vehicle crossflow drag coefficient, and r_{mean} is the mean fin height above the vehicle centreline.

6.5.4 Calculated Coefficients

The first test of the script for coefficient calculation was carried out by recalculating the parameters of a REMUS 100 vehicle as in [66]. The results and comparison with the original parameters calculated can be seen in Table 6.4. An error was found in respect to N_{rr} as it should be equal to M_{qq}, the other values with a high difference are M_{ww} and N_{ur}. The equation has been checked but the calculated values do not match those described in [66]. This will require examination after more field tests with the vehicle.

Table 6.4: Remus 100 coefficients comparison

Parameter	Remus Preterus	Remus calculated	Difference
Xuu	-1.62	-1.62	0%
Xwq	-3.55E+01	-3.55E+01	0%
Xqq	-1.93	-2.038	6%

Continued on next page

Table 6.4 – *Continued from previous page*

Parameter	Remus Preterus	Remus calculated	Differerence
Xvr	3.55E+01	3.55E+01	0%
Xrr	-1.93	-2.038	6%
Yvv	-1.31E+02	-1.30E+02	-1%
Yrr	6.32E-01	1.08E+00	71%
Yuv	-2.86E+01	-3.22E+01	13%
Ywp	3.55E+01	3.55E+01	0%
Yur	5.22	5.69E+00	9%
Ypq	1.93	2.038	6%
Zww	-1.31E+02	-1.30E+02	-1%
Zqq	-6.32E-01	-1.08E+00	71%
Zuw	-2.86E+01	-3.22E+01	13%
Zuq	-5.22	-5.6922	9%
Zvp	-3.55E+01	-3.55E+01	0%
Zrp	1.93	2.038	6%
Yuudr	9.64	10.3624	7%
Nuudr	-6.15	-6.6112	7%
Zuuds	-9.64	-10.3624	7%
Kpp	-1.30E-01	-8.40E-03	-94%
Kpdot	-7.04E-02	8.90E-03	-113%
Mww	3.18	-6.38	-301%
Mqq	-1.88E+02	-1.71E+02	-9%
Mrp	4.86	4.91E+00	1%
Muq	-2	6.26E+00	-413%
Muw	2.40E+01	3.35E+01	40%
Mwdot	-1.93	-2.04E+00	6%
Mvp	-1.93	-2.04E+00	6%
Muuds	-6.15	-6.61E+00	7%
Nvv	-3.18	6.38E+00	-301%
Nrr	-9.40E+01	-1.71E+02	82%
Nuv	-2.40E+01	-3.56E+01	48%
Npq	-4.86	-4.9071	1%
Ixx	1.77E-01	1.77E-01	0%
Iyy	3.45	3.45	0%
Izz	3.45	3.45	0%
Nwp	-1.93	-2.038	6%
Nur	-2	6.256	-413%

Continued on next page

Table 6.4 – *Continued from previous page*

Parameter	Remus Preterus	Remus calculated	Differerence
Xudot	-9.30E-01	-9.19E-01	-1%
Yvdot	-3.55E+01	-3.55E+01	0%
Nvdot	1.93	2.038	6%
Mwdot	-1.93	-2.038	6%
Mqdot	-4.88	-4.8982	0%
Zqdot	-1.93	-2.038	6%
Zwdot	-3.55E+01	-3.55E+01	0%
Yrdot	1.93	2.038	6%
Nrdot	-4.88	-4.8982	0%

With the results from the comparison with the original values of Remus 100 the coefficients for MULLAYA AUV are calculated. The final calculated coefficients can be seen in Table 6.5.

6.6 Model Validation Experiment Setup

During the week of 22–26 October 2018, MULLAYA AUV was the subject of a small series of test to capture sensors' data and vehicle position in the AMC's Survival Centre pool. For each test run, the vehicle measured the heading, roll and pitch from the compass; x,y,z acceleration and roll, yaw and pitch from the inertial motion unit; total current; propeller RPM; depth; and control data logging. Each of these variables had a timestamp and in the case of sensors were sampled at 10 Hz in the main CPU. The electronic characteristics of MULLAYA can be seen in Table 6.6.

The propeller RPM was controlled with a PID on an Arduino Leonardo, and the microcontroller (MCU) received the measurement from a hall effect sensor located near the propeller shaft. Over the shaft, a series of neodymium magnets were attached to produce the required magnetic field. This arrangement replaces the old variable reluctance sensor, allowing more accurate measurement at low speed. The MCU applies an average filter over five vehicle samples to avoid an abrupt answer from the PID. This

Table 6.5: MULLAYA AUV coefficients

Parameter	**Result**	**Parameter**	**Result**	**Parameter**	**Result**
Xuu	-2.7969	Zrp	0.6789	Nuv	-30.8809
Xwq	-29.6403	Yuudr	12.1167	Npq	-5.0635
Xqq	-0.6789	Nuudr	-7.5123	Ixx	0.08394602
Xvr	29.6403	Zuuds	-12.1167	Iyy	3.08444574
Xrr	-4.9481	Kpp	-1.30E-01	Izz	3.08444574
Yvv	-95.3687	Kpdot	1.09E-02	Nwp	0.6579
Yrr	-2.4529	Mww	6.9559	Nur	-5.3156
Yuv	-32.937	Mqq	-135.0375	Xudot	-0.512
Ywp	29.6403	Mrp	5.0635	Yvdot	-29.6403
Yur	7	Muq	-5.3366	Nvdot	0.6579
Ypq	0.6789	Muw	27.1576	Mwdot	0.6579
Zww	-95.3687	Mwdot	-0.6789	Mqdot	-5.0526
Zqq	2.4529	Mvp	-0.6789	Zqdot	-4.9481
Zuw	-32.937	Muuds	-7.7304	Zwdot	-29.6403
Zuq	-7	Nvv	6.9559	Yrdot	4.9481
Zvp	-29.6403	Nrr	-135.0375	Nrdot	5.0526

Table 6.6: MULLAYA vehicle electronic system components

Component	**Application**
Lattepanda	Main CPU
Arduino Leonardo 1	Stepper Controller
Arduino Leonardo 2	Propeller Controller
HMR 3000	Compass
IMU 440	Inertial Motion Unit
ACS712 Module	Current Sensor
Artou Hall sensor	RPM sensor
Keller pressure sensor	Depth sensor

average measurement is reported back to the main CPU. The main CPU provides the three control signals: rudder angle, and elevator angle and propeller RPM, these signals come from a simple script that provides a fix desire RPM and two PIDs. The depth and heading were controlled with these PIDs without taking into consideration the vehicle mathematical model. Both heading and depth PID were partially calibrated such that the vehicle was able to run the length of the pool without crashing into the sides. The objectives of these experiments were: to capture real field data from the vehicle sensors, test the semi-empirical calculated vehicle model, and capture data from the pressure sensors in the head. Full details of the experiment conducted can be seen in Table 6.7.

Table 6.7: List of experiments and files name

#	Date	Time	Propeller File	Fins File	Sensor data file
Day 1					
1	22/10/2018	15.36	DATA∼ 15.txt	DATA∼ 15.txt	DATA_logging10.txt
2	22/10/2018	15.45	DATA∼ 16.txt	DATA∼ 16.txt	DATA_logging11.txt
3	22/10/2018	15.51	DATA∼ 17.txt	DATA∼ 17.txt	DATA_logging12.txt
4	22/10/2018	15:57	DATA∼ 18.txt	DATA∼ 18.txt	DATA_logging13.txt
5	22/10/2018	16:05	DATA∼ 19.txt	DATA∼ 19.txt	DATA_logging14.txt
Day 2					
1	23/10/2018	10:43	DATA∼4.txt	DATA∼4.txt	DATA_logging3.txt
2	23/10/2018	10:57	DATA∼5.txt	DATA∼5.txt	DATA_logging4.txt
3	23/10/2018	11:12	DATA∼6.txt	DATA∼6.txt	DATA_logging5.txt
4	23/10/2018	11:30	DATA∼7.txt	DATA∼7.txt	DATA_logging6.txt
5	23/10/2018	11:36	DATA∼8.txt	DATA∼8.txt	DATA_logging7.txt
6	23/10/2018	14:33	DATA∼9.txt	DATA∼9.txt	DATA_logging8.txt
Day 3					
No experiments					
Day 4					
1	25/10/2018	10:58	DATA∼12.txt	DATA∼12.txt	DATA_logging14.txt
20	25/10/2018	11:03	DATA∼13.txt	DATA∼13.txt	DATA_logging15.txt
19	25/10/2018	11:12	DATA∼14.txt	DATA∼14.txt	DATA_logging16.txt
18	25/10/2018	11:17	DATA∼15.txt	DATA∼15.txt	DATA_logging17.txt
17	25/10/2018	11:22	DATA∼16.txt	DATA∼16.txt	DATA_logging18.txt
16	25/10/2018	11:31	DATA∼18.txt	DATA∼18.txt	DATA_logging19.txt
15	25/10/2018	11:37	DATA∼19.txt	DATA∼19.txt	DATA_logging20.txt

Continued on next page

Table 6.7 – *Continued from previous page*

#	Date	Time	Propeller File	Fins File	Sensor data file
14	25/10/2018	11:41	DATA~20.txt	DATA~20.txt	DATA_logging21.txt
13	25/10/2018	11:45	DATA~21.txt	DATA~21.txt	DATA_logging22.txt
12	25/10/2018	11:48	DATA~22.txt	DATA~22.txt	DATA_logging23.txt
11	25/10/2018	11:52	DATA~23.txt	DATA~23.txt	DATA_logging24.txt
10	25/10/2018	11:55	DATA~24.txt	DATA~24.txt	DATA_logging25.txt
9	25/10/2018	11:59	DATA~25.txt	DATA~25.txt	DATA_logging26.txt
8	25/10/2018	12:02	DATA~26.txt	DATA~26.txt	DATA_logging27.txt
7	25/10/2018	12:08	DATA~27.txt	DATA~27.txt	DATA_logging28.txt
6	25/10/2018	12:12	DATA~28.txt	DATA~28.txt	DATA_logging29.txt
5	25/10/2018	12:16	DATA~29.txt	DATA~29.txt	DATA_logging30.txt
4	25/10/2018	12:20	DATA~30.txt	DATA~30.txt	DATA_logging31.txt
3	25/10/2018	12:24	DATA~31.txt	DATA~31.txt	DATA_logging32.txt
2	25/10/2018	12:27	DATA~32.txt	DATA~32.txt	DATA_logging33.txt
1	25/10/2018	12:32	DATA~33.txt	DATA~33.txt	DATA_logging34.txt

To capture the vehicle's real position, a stereo camera system was developed. The stereo system was composed of two GoPro HERO4s mounted over a 3D printed frame. This mount was located on the side of the pool. A script was written that can capture the location of the centre of a yellow mark and report the location. Two yellow marks were placed on the vehicle, as can be seen in Figure 6.4. Two marks were located to allow not only the calculation of $< x, y, z >$ location but also pitch and yaw. The post- processing algorithm can be seen in Algorithm 6.1. This calibration is undertaken by taking a

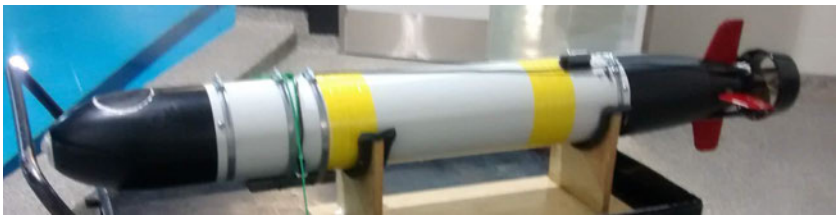

Figure 6.4: MULLAYA with stereo marks

series of pictures of a chequerboard pattern Figure 6.5 wAfter the acquisition of the image, the application of MATLAB for stereo camera calibration was employed with

input : Stereo calibration, video left, video right

output: $< x(t), y(t), z(t) >$

while *frame available* **do**

$Load\ video\ left$;

$Load\ video\ right$;

$Correct\{Stereocalibration, frameleft, frameright\}$;

$< xm, ym > \leftarrow DetectYellow(frameleft)$;

$DisparityMap \leftarrow DisparityMapBuild\{frameleft, frameright\}$;

$< x(t), y(t), z(t) > \leftarrow Positioncalculator\{DisparityMap, < xm, ym >\}$;

end

Algorithm 6.1: Stereo camera post processing algorithm

Figure 6.5: Stereo image calibration example

the results shown in Figure 6.6 and Figure 6.7.

An example of the measured kinematic acceleration data obtained from the sensors in one of the trial can be seen in Figures 6.8 to 6.10. These data were obtained by subtracting the gravity acceleration from the measured acceleration.

6.7 Experiment Results

The calculated mathematical model of MULLAYA AUV was simulated with the same inputs of the pool experiments. The Simulink equivalent model contains a first-order hold function and a delay function of $0.3s$ to represent the delay of the communication between MATLAB and the vehicle actuator (Figure 6.11). Intermediate data between the measure points are calculated as the linear interpolation between the measured points. Figures 6.12 to 6.14 shows the employed command signals that feed MULLAYA

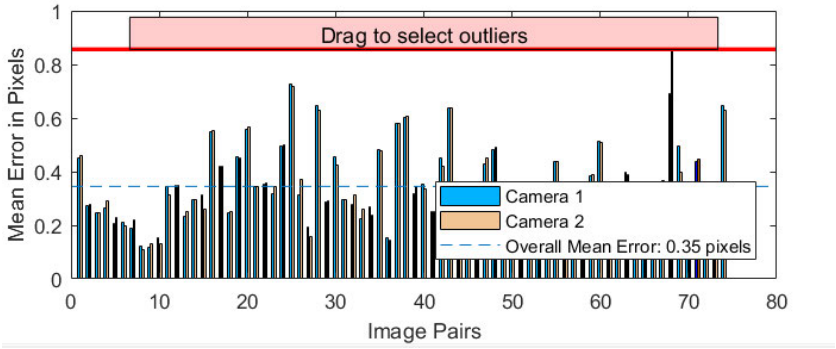

Figure 6.6: Reprojection errors

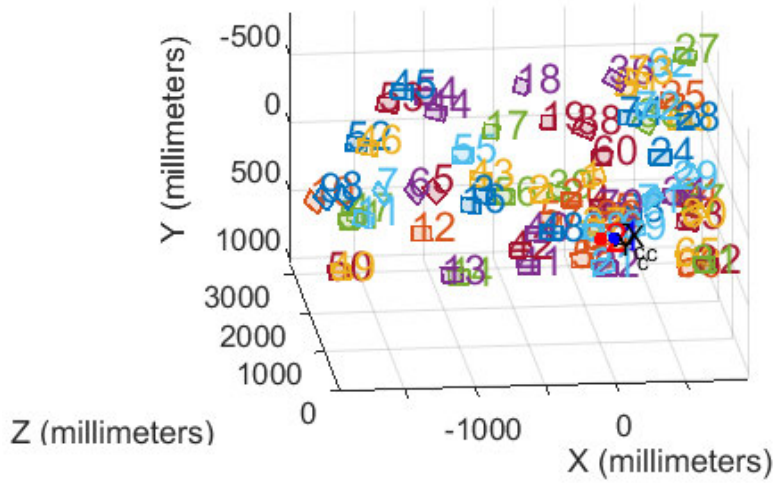

Figure 6.7: Extrinsic pattern view

model. A total of 20 seconds were simulated with data having been captured each 0.1s.

The propeller of MULLAYA forced the vehicle into a 5 degrees angle of roll as a reaction to the force and moment produced by the propeller. The vehicle at 1000 RPM showed on the experiment an approximate speed to 1 m/s at stable RPM. The values of the propulsion model were set such that they matched these values. The results from the vehicle simulation and the experiment roll values can be seen in Figures 6.15 and 6.16.

Figures 6.17 to 6.19 present the comparative results from the camera measurement and the simulation of the vehicle. The data employed for this were from experiment 5 of day 1. The displacement measured with the stereo cameras was similar to the values obtained from the simulations. Figure 6.20 shows the result from post-processing the

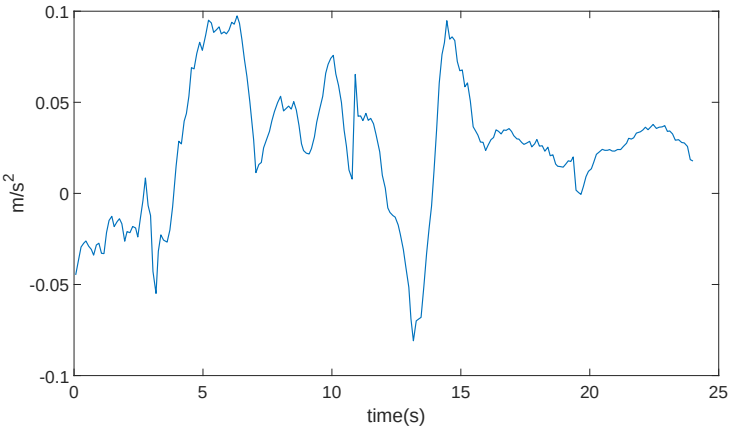

Figure 6.8: Kinematic acceleration in surge direction

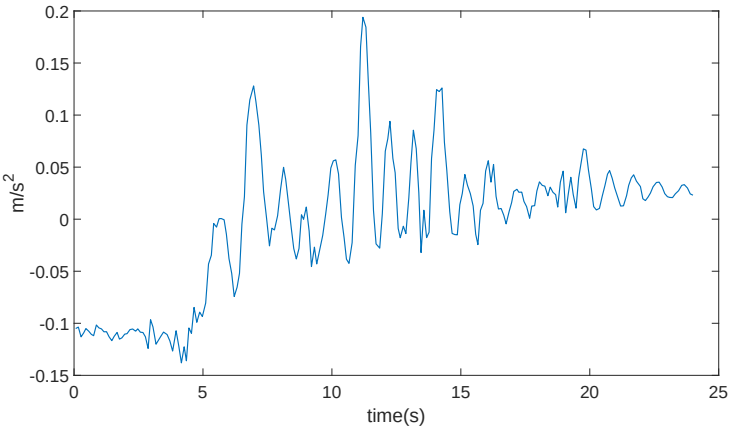

Figure 6.9: Kinematic acceleration in heave direction

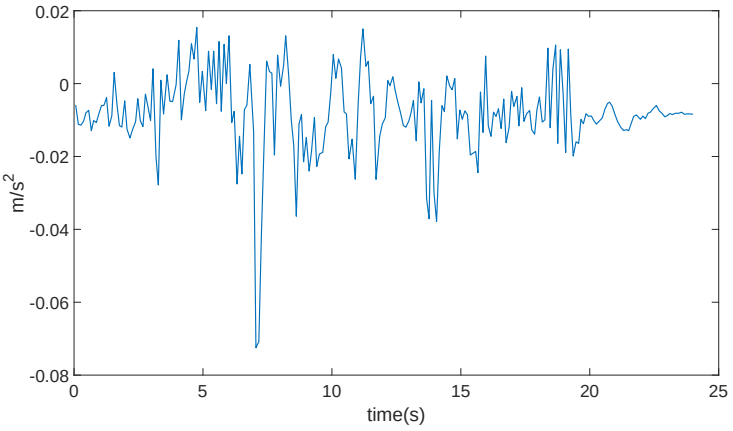

Figure 6.10: Kinematic acceleration in sway direction

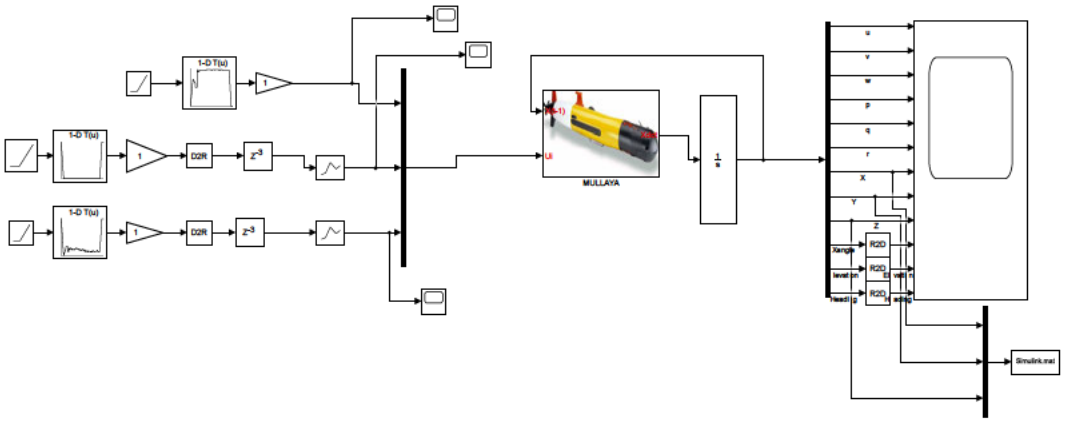

Figure 6.11: Simulink model MULLAYA with experimental data

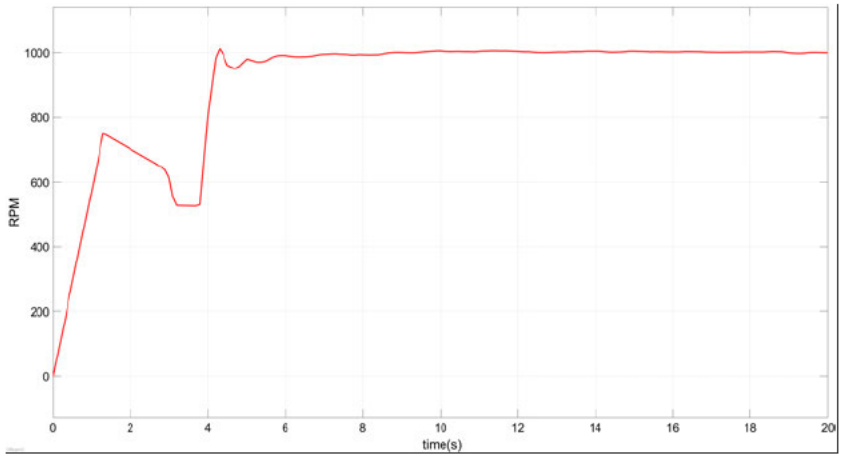

Figure 6.12: Propeller signal from an experiment used in simulink simulation

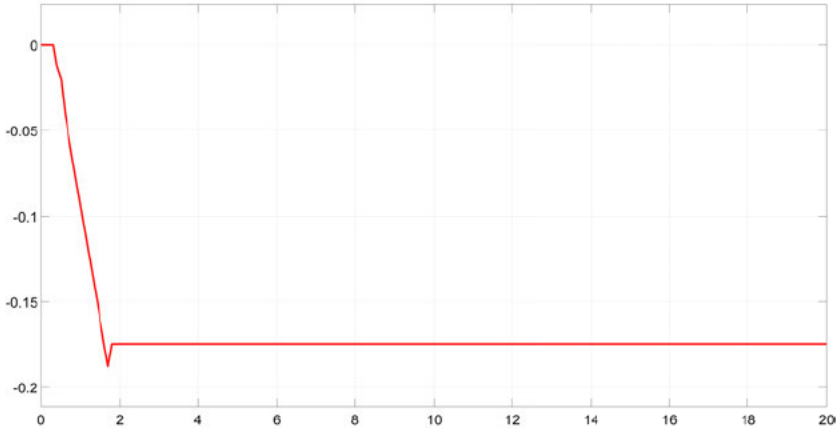

Figure 6.13: Elevator signal from an experiment used in simulink simulation

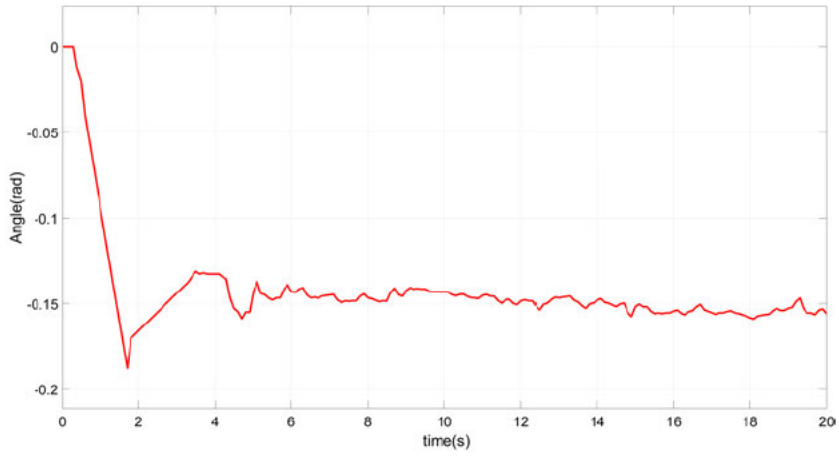

Figure 6.14: Rudder signal from an experiment used in Simulink simulation

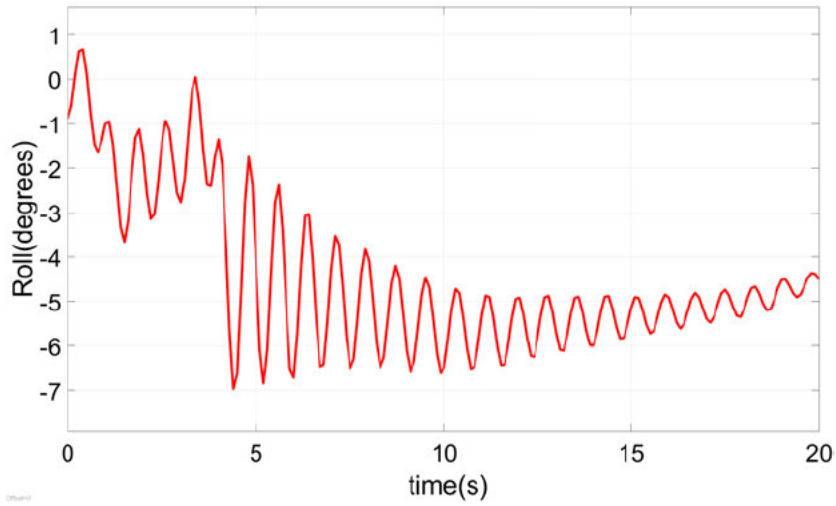

Figure 6.15: Roll result from Simulink model

measurement from the depth sensor. The results from the measurement of the stereo cameras suffered from high noise as the illumination affected the creation of the disparity map. Although there is a similarity between the calculated model and the vehicle model.

6.8 Conclusions

The main challenge in confirming the mathematical model of MULLAYA AUV was that the vehicle does not reach a stable speed. However, the results obtained show

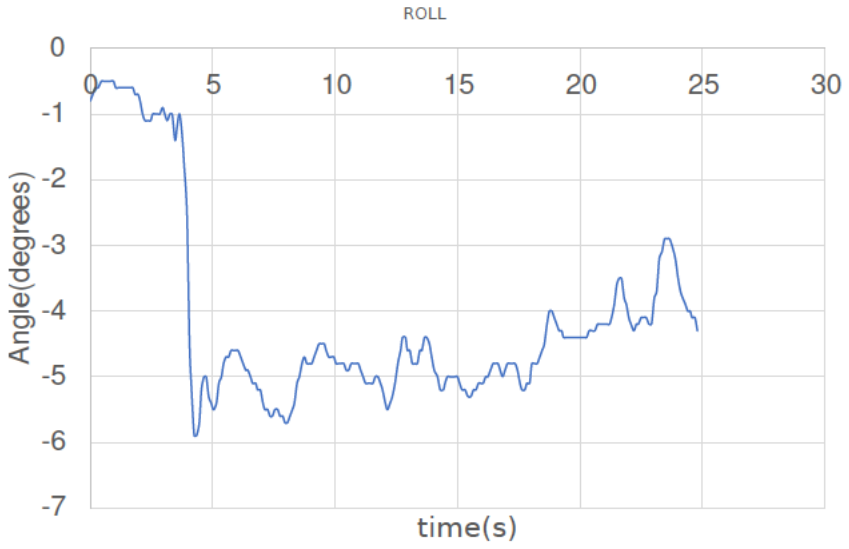

Figure 6.16: Roll result from experiment

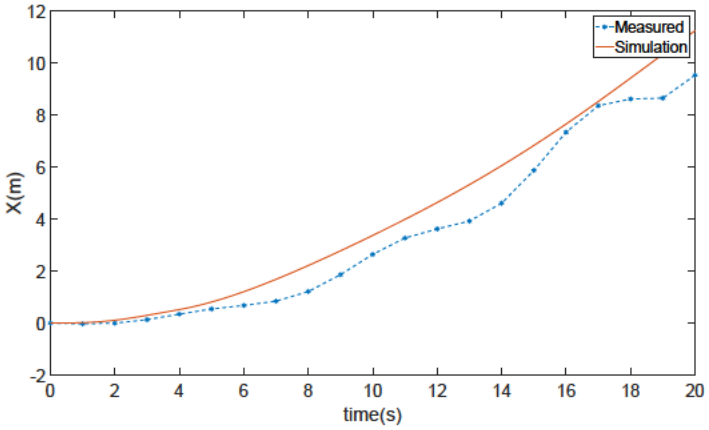

Figure 6.17: X result from experiment and simulation

that semi-empirical calculations of hydrodynamic coefficients are a valid methodology for approximation of the coefficients from underwater vehicles. In the experiments, the sources of noise that add error between the mathematical model and the experimental measurement cannot be completely established. The technique of measurement with stereo cameras included noise at the calculation of the disparity map, as many frame pixels could not establish their location. In the inverse, the error introduced by the mathematical modes is because the shape of MULLAYA AUV employed in the experiments differs lightly in dimensions from the employed vehicle on the calculation of the

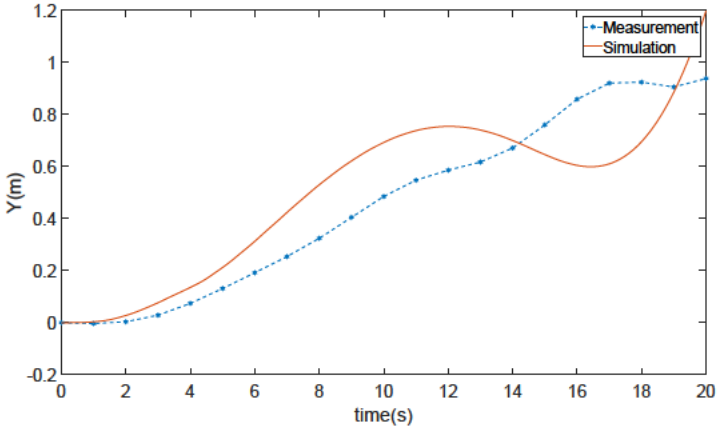

Figure 6.18: Y result from experiment and simulation

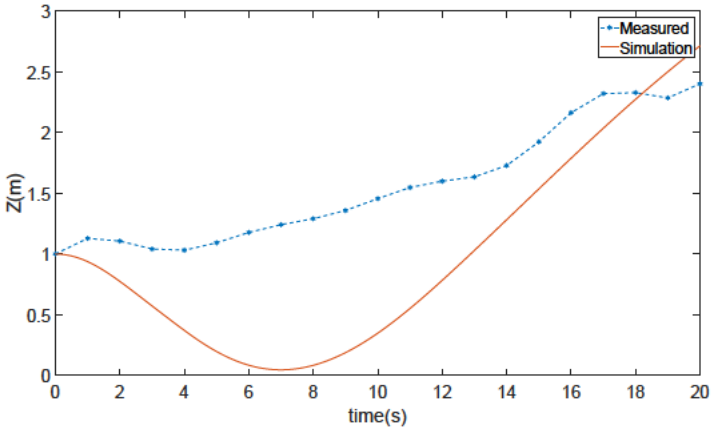

Figure 6.19: Z result from experiment and simulation

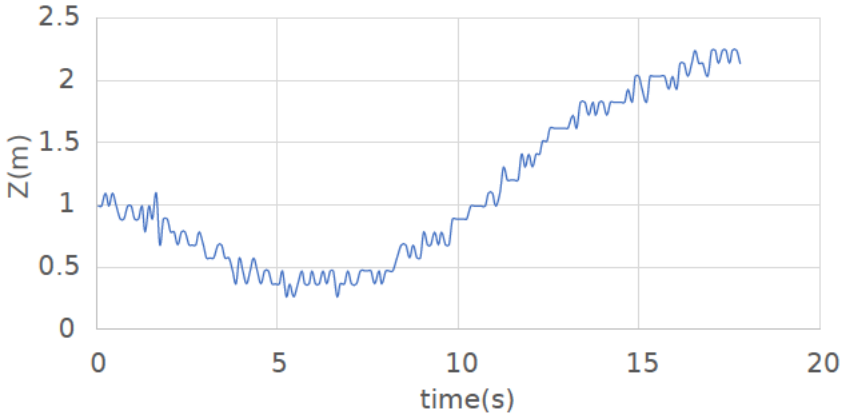

Figure 6.20: Z result from experiment measured with depth sensor

coefficients. In this study, an example is that the propeller duct was not included and its effects over the coefficients. However, the range of differences between the mathematical model and the real experiments were expected as semi-empirical methods are only an approximation and more experiments are required to establish more precise vehicle coefficients. Nonetheless, the calculated mathematical model can be used to calculate and simulate motion controllers.

This chapter has described the calculation of a mathematical model for MULLAYA underwater vehicle in six degrees of freedom and its comparison with previous similar vehicles and a series of experiments that were carried out. In this model, the external forces and moments resulting from hydrostatics, hydrodynamic lift and drag, added mass, and the control inputs of the vehicle propeller and fins are all defined in terms of vehicle coefficients. The calculation of the coefficients was conducted by replicating the work done [66] by creating a script capable of calculating the coefficients. This model will be employed in Chapter 7 for the learning of policies for motion control of MULLAYA AUV. The calculation of the coefficients for an underwater vehicle can be carried out in several ways. Experimental techniques such as planar motion mechanics isolate particular characteristics to allow the calculation of coefficients, vehicle trials paired with observers as the Kalman filter allow the calculation of the coefficients. Other modern common techniques are the use of CFD. Although CFD methodology has gained popularity, there is a dependency on the correct user input [110]. Other methods commonly used are the semi-empirical methodologies, and semi-empirical approaches use experimentally derived guidelines for estimating model parameter values for vehicles with generic shapes. Analytical methods for calculation of model parameter as strip theory or solving Laplace's equation can be implemented. Strip theory, also known as slender body approximation, can estimate the hydrodynamic coefficients for a body using 2D sectional properties. Strip theory can also approximate other coefficients in the equations of motion, such as damping coefficients [111].

In this chapter a combination of semi-empirical techniques has been used for the calculation of the coefficients, and verification of the methodology was undertaken by replicating calculation from previous work and by simulating the vehicle behaviour and comparing it to a series of experiments. The equations determining the coefficients, as well as

those describing the vehicle rigid-body dynamics, were left in non-linear form to better simulate the inherently non-linear behaviour of the vehicle.

Chapter 7

Exploration of the Applicability of Probabilistic Inference for Learning Control on Underactuated Autonomous Underwater Vehicles

At the time of submission of this thesis the present chapter is under review by the *Journal of Autonomous Robots*. This chapter employs the mathematical model of MULLAYA AUV calculated in chapter 6.

7.1 Introduction

Autonomous underwater vehicles play an important role in the exploration of the seas. This exploration is primarily driven by commercial, military and scientific needs. In this context, the proper and correct navigation of the vehicle is a key requirement. Motion controllers that are used for navigating AUVs can be classified into four basic strategies: point stabilisation [119], trajectory tracking [120], path following [2] and path tracking [121]. Point stabilisation controllers stabilise a vehicle to the desired goal posture from an initial configuration [122]. Trajectory tracking controllers use a virtual vehicle to generate a reference trajectory that has an associated time required to be employed by the real vehicle [123]. In the case of path following the vehicle is forced to pursue the desired path without temporal specifications. With regard to the path following controller, Frenet-Serret line-of-sight (LOS) is usually employed, coupled with another controller to minimise the error between the obtained geometric references and the vehicle variables. The final strategy is path tracking, which combines trajectory tracking and path following by the introduction of a virtual time parameter to force the vehicle to complete the path within a specific time.

Difficulty in controlling underwater vehicles arises due to the non-linear and time-varying dynamics of underwater vehicles, uncertainties in their hydrodynamic coefficients and disturbance in the environment (e.g. ocean currents). Furthermore, all complexities are exacerbated for the controller in underactuated vehicles [122]; underactuated vehicles have more degrees of freedom to be controlled than surfaces of control. Nevertheless, this configuration is more prevalent as it is the most energy efficient design for travelling at high speeds [124].

Waypoint tracking is the most common methodology to control a vehicle, e.g. commercial vehicles such as Gavia [125] and REMUS [126] use this methodology. Waypoint tracking is directing the vehicle to approximate to a series of specific target points. The vehicle calculates the required direction to which the vehicle should be directed and upon arriving at the proximities of a point is given a new target. Like many industries, PID is the most common methodology to control the vehicle orientation and speed. However, there is research to employ more robust options than PID. [127] employs a

NARMAX model from the vehicle and a constrained self-tuning controller to direct the vehicle to the respective target. Other methodologies employ a combination of LOS for waypoint [128] and a standard controller or backstepping techniques to minimise the error between the vehicle position and the desired position [129].

In the case of LOS, commercially LOS-PID and LOS-Fuzzy controllers are employed as their implementation are simpler and more accepted in the industry [108, 130, 131]. Other research for example [132] used an LOS guidance law with two integrators and three feedback controllers to compensate for external unknown perturbation such as ocean current which is one of the weaknesses for methodologies such as PID/Fuzzy controllers. [133] have used an alternative methodology of a grey prediction to obtain the next AUV position in advance and then use LOS to calculate the desired angles, such that if there is environmental interference, the vehicle will not be affected.

In the search for more robust controllers, nonlinear control techniques haev been explored. [134] have designed a horizontal path following controller based on Lyapunov stability theorem and backstepping method. In [135], a method consisted of Lyapunov stability theorem and feedback gain backstepping reduce the complexity of the controller and improve adjustability of the parameters. Another methodology proposes a global path following for AUVs based on the same coordinates to achieve global asymptotic stability of the following error [136]. Following the research into backstepping, [8] have adopted fuzzy backstepping sliding mode control to overcome non-linearities, uncertainties and external disturbances.

However, the aforementioned research in controls are focused to provide a more robust path following performance. The controller still requires the calibration of parameters or specific design of observers to identify the unknown parameters of the dynamic model. A methodology to overcome this is the use of machine learning algorithms. The most prominent algorithm in machine learning is neural networks. In particular, the research to control underwater vehicles had focused on the use of machine learning algorithms to recognise uncertainties. [137, 138] have designed a combined version of control law for the convergence of the kinematic model and an adaptive backstepping sliding control based in radial basis function (RBF) neural network to identify the unknown parameters

of the dynamic model. In a similar way, [9] have reduce the backstepping complexity by the inclusion of a second-order filter to obtain the derivatives of the virtual controller and filter high-frequency measurement noise, and coupled the filter with an RBF neural network that compensates for vehicle uncertainties.

Although the practicality of machine learning has been largely to identify uncertainties in AUV control, some machine learning algorithms are capable of doing more such as controlling the vehicle directly. Recently, there has been increasing research efforts into the use of reinforcement learning to generate policies to control underwater vehicles and robots in general. An example of machine learning control can be seen in [139],where reinforced learning (RL) based on the Markov decision process (MDP) was employed to produce a policy capable of controlling a vehicle around an obstacle with minimum cost. In the case of path following, reinforced learning had been applied to path following of ships. In [140], an actor-critic multilayer perception reinforced learning is used to reduce the tracking error to zero. Deep reinforced learning has also been proposed as a possible solution for the tracking problem. [141] employed two neural networks. The primary neural network selects the action and the secondary evaluates whether the produced action is valid; with further modification through a deep deterministic policy gradient. Another application used continuous actor-critic learning automaton algorithm to teach an AUV to follow a pipeline [142], considering the improved performance in search of the policy of this algorithm the number of episodes over the platform can be in the hundreds.

RL can be divided into two methodologies: model-based methods and model-free methods, such as Q-learning [143] or TD-learning [144]. The application of path following control based in traditional RL such as Q-learning [145] is highly complex and difficult as a high quantity of experiments is required to acquire data and test each policy iteration. The additional difficulties in underwater vehicles are vehicle safety, maximum time underwater and computational power. RL for a system with low-dimensional state spaces and fairly favourable dynamics can require thousands of trials to arrive at the appropriate policies[141, 146].

Model-based RL methods are more efficient than model-free methods in searching for a

useful policy, as the policy is searched over a model and not the real platform. However, their accuracy can suffer severely from model errors. A solution to address the model errors is the use of probabilistic models to express its uncertainty. An application of model-free methodologies that use a probabilistic methodology was proposed by [147]. Their methodology use an on-line selective reinforcement learning approach combined with Gaussian process (GPs) regression for learning reference tracking control policies given no prior knowledge of the dynamical system. [146] have proposed Probabilistic Inference for Learning Control (PILCO), which is a model-based policy search method. The probabilistic model uses non-parametric Gaussian processes (GPs) to characterise the model uncertainty and the policy improvement is based on analytic policy gradients which employs deterministic approximate interference techniques. Due to probabilistic modelling and inference approach, PILCO can achieve higher learning efficiency than other methods in continuous state-action domains and, hence, is directly applicable to complex mechanical systems, such as robots.

The current research explores the applicability of PILCO to control underactuated AUVs by a series of simulations with different objectives and target values. The main goals of the implementation of reinforced learning with PILCO are:

- Minimum quantities of episodes over the platform;
- Small test time over the platform;
- Minimum quantity of variables to be predicted by the GP; and
- Vehicle safety.

7.2 Underwater Vehicle Mathematical Model

In Fossen et al. [21] it was shown that the non-linear dynamic equations of motion of an underwater vehicle can be expressed in vector notation defined by a state vector composed by the vector v of velocities on the body frame of the form $[u, v, w, p, q, r]^T$ and the vector η of position in the earth fixed frame (Figure 7.1) of the form $[\xi, \eta, \zeta, \phi, \theta, \psi]^T$ such that

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\,\mathbf{v} + \mathbf{D}(\mathbf{v})\,\mathbf{v} + \mathbf{g}(\eta) = \tau \tag{7.1}$$

with the kinematic equation

$$\dot{\eta} = \mathbf{J}(\eta)\,\mathbf{v} \tag{7.2}$$

where

η position and orientation of the vehicle in earth-fixed frame,

$\mathbf{v}$ linear and angular vehicle velocity in body fixed frame,

$\dot{\mathbf{v}}$ linear and angular vehicle acceleration in body fixed frame,

$\mathbf{M}$ matrix of inertial terms,

$\mathbf{C}(\mathbf{v})$ matrix of Coriolis and centripetal terms,

$\mathbf{D}(\mathbf{v})$ matrix consisting of damping or drag terms,

$\mathbf{g}(\eta)$ vector of restoring forces and moments due to gravity and buoyancy,

τ vector of control and external forces, and

$\mathbf{J}(\eta)$ rotation matrix that converts velocity in a body fixed frame v to an Earth fixed frame velocity $\dot{\eta}$.

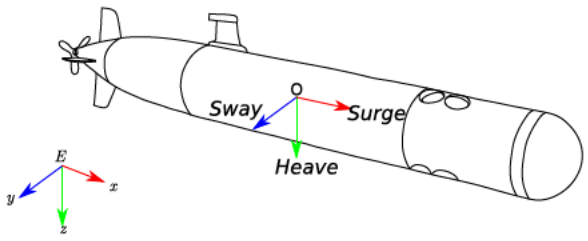

Figure 7.1: AUV different reference frames, vehicle frame is equal to the centre of buoyancy.

Equation (7.1) can be expanded into a more general equation of motion as has been shown in [66, 67]. The result of the expansion will be a system of six equations with 73 hydrodynamic coefficients. However, for a complete model the control surfaces must be modelled. In a general case, the resulting forces and moments of a control surface

(thrusters and fins) can be expressed as [21];

$$\begin{aligned} F_{prop} &= -K_{fprop}\,|n|\,n \\ M_{prop} &= -K_{mprop}\,|n|\,n \end{aligned} \tag{7.3}$$

$$\begin{aligned} L_{fin} &= K_{L|\delta_{\text{fin}}}\delta_{\text{fin}}v_e^2 \\ M_{fin} &= K_{M|\delta_{\text{fin}}}\delta_{\text{fin}}v_e^2 \end{aligned} \tag{7.4}$$

A more accurate thruster model can be found in [68] with the inclusion of the motor model and fluid dynamics. However, in this study, the more conservative model from Fossen et al. [21] is used.

7.3 LOS Guidance Law Mathematical Background

This section describes the mathematical background of the 3D guidance law employed in the present study for control of an underactuated AUV. In this study, it was decided to employ the LOS proposed in [108] as it is an extension of the work of [21].

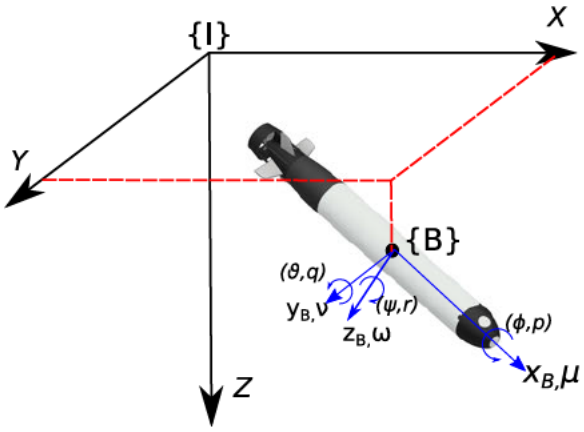

Figure 7.2: LOS reference frames

If the vehicle kinematic is represented by its spatial position $p(t) \triangleq [x(t), y(t), z(t)]^T$ and its velocity is represented by $v(t) \triangleq \dot{p}(t) \in \mathbb{R}^3$ state is related to the $\{I\}$ frame. Also, the speed is represented by $U(t) \triangleq |v(t)| = \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2 + \dot{z}(t)^2} > 0$. The steering of the underactuated AUV is characterised by the azimuth angle χ and elevation v

(Figure 7.2).

$$\begin{cases} \chi = \operatorname{atan2}(\dot{y}, \dot{x}) \\ \upsilon = \arctan\left(\frac{-\dot{z}}{\sqrt{\dot{x}^2+\dot{y}^2}}\right) \end{cases} \tag{7.5}$$

If it is considered a continuous path parametrised by a scalar variable $\varpi \in \mathbb{R}$, the position of a point over the path is represented by $p_p(\varpi) \in \mathbb{R}^3$ (Figure 7.3). Similarly, the orientation of the point can be defined as

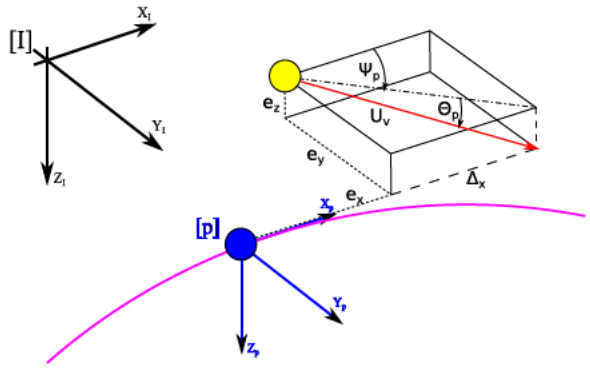

Figure 7.3: LOS main variables

$$\begin{cases} \psi_p = \operatorname{atan2}(y'_p, x'_p) \\ \theta_p = \arctan\left(\frac{-z'_p}{\sqrt{x'^2_p+y'^2_p}}\right) \end{cases} \tag{7.6}$$

where $x'_p = dx_p/d\varpi$, $y'_p = dy_p/d\varpi$ and $z'_p = dz_p/d\varpi$. Hence the tracking error is expressed as:

$$\varepsilon = [x_e, y_e, z_e]^{\mathrm{T}} = \mathbf{R}_F^{\mathrm{T}}(p - p_p) \tag{7.7}$$

where $\mathbf{R}_F^{\mathrm{T}} := \mathbf{R}_z(\psi_p)\,\mathbf{R}_y(\theta_p)$

$$\mathbf{R}_z = \begin{bmatrix} \cos(\psi_p) & -\sin(\psi_p) & 0 \\ \sin(\psi_p) & \cos(\psi_p) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.8}$$

$$\mathbf{R}_z = \begin{bmatrix} \cos(\theta_p) & 0 & \sin(\theta_p) \\ 0 & 1 & 0 \\ -\sin(\theta_p) & 0 & \cos(\theta_p) \end{bmatrix} \tag{7.9}$$

If the following Lyapunov control function positive definitive is

$$V_{\varepsilon} = \frac{1}{2}\varepsilon^{T}\varepsilon \tag{7.10}$$

The derivative of equation (7.10) can be written as

$$\begin{aligned} V_{\varepsilon} = x_e \left(U_d \cos\left(\psi_r\right) \cos\left(\theta_r\right) - U_p\right) + \\ y_e U_d \sin\left(\psi_r\right) \cos\left(\theta_r\right) - z_e U_d \sin\left(\theta_r\right) \end{aligned} \tag{7.11}$$

where U_d is the desired composite speed of the AUV. The auxiliary control input of the virtual point P_p is chosen as:

$$U_p = U_d \cos(\psi_r) \cos(\theta_r) + k_x x_e \tag{7.12}$$

where the steering angles are:

$$\begin{cases} \psi_r = \arctan\left(\frac{-k_y y_e}{\Delta_y}\right) \\ \theta_r = \arctan\left(\frac{k_z z_e}{\Delta_z}\right) \end{cases} \tag{7.13}$$

If the guidance variables Δ_y and $\Delta_z > 0$, the control gains k_x, k_y, k_z are positive constants. If equation (7.12) and equation (7.13) are substituted into equation (7.11) and considering the relationship among inertial frame $\{I\}$, flow frame $\{W\}$ and path frame $\{F\}$, the desired azimuth angle υ_d and elevation angle χ_d can be written as [148]:

$$\begin{aligned} \upsilon_d &= \arcsin(\sin\theta_p \cos\psi_r \cos\theta_r + \cos\theta_p \cos\theta_r) \\ \chi_d &= atan2(\chi_{dy}, \chi_{dx}) \end{aligned} \tag{7.14}$$

where

$$\begin{aligned} \chi_{dy} = \cos\psi_p \sin\psi_r \cos\theta_r - \sin\psi_p \sin\theta_p \sin\theta_r \\ + \sin\psi_p \cos\theta_p \cos\psi_p \cos\theta_r \end{aligned} \tag{7.15}$$

$$\begin{aligned} \chi_{dx} = -\sin\psi_p \sin\psi_r \cos\theta_r - \cos\psi_p \sin\theta_p \sin\theta_r \\ + \cos\psi_p \cos\theta_p \cos\psi_p \cos\theta_r \end{aligned} \tag{7.16}$$

In order to transform the path following to path tracking the path was defined over time together with equation (7.14) and equation (7.12). This was to produce not only the desire angles but also the required speed at each time instance. The steering error vector can be expressed as $[e_\mu, e_\upsilon, e_\chi]$ where $e_\mu = \mu_d - \mu_v$, $e_\upsilon = \upsilon_d - \upsilon_v$ and $e_\chi = \chi_d - \chi_v$.

7.4 Probabilistic Inference for Learning Control (PILCO)

PILCO algorithm [146, 149] (Algorithm 7.1) employs GPs as the base for policy search. A GP can be defined by a mean function $m(\cdot)$ and a positive definitive covariance function $k(\cdot,\cdot)$ commonly known as kernel. Usually a prior mean function $m \equiv 0$ and an exponentiated quadratic kernel (equation (7.17)) are employed. This kernel only has two parameters to learn, l that determines the length of the 'wiggles' in the function and σ^2 which determines the average distance of the function away from its mean [150].

$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) \tag{7.17}$$

Given n training inputs $X = [x_1, ..., x_n]$ and corresponding training targets $Y = [y_1, ..., y_n]$, the posterior GP hyper-parameters l and σ^2 are learned by evidence maximisation [43]. The posterior predictive distribution $p(f_*|x_*)$ of the function value $f_* = f(x_*)$ for a test input x_* is Gaussian with mean and variance

$$\begin{aligned} m_f(x_*) &= \mathbf{k}_*^T \left(\mathbf{K} + \sigma_\varepsilon^2 \mathbf{I}\right)^{-1} y \\ \sigma_f^2(x_*) &= k_{**} - \mathbf{k}_*^T \left(\mathbf{K} + \sigma_\varepsilon^2 \mathbf{I}\right)^{-1} k_* + \sigma_\varepsilon^2 \end{aligned} \tag{7.18}$$

[1] ***init:*** *Sample controller parameters* $\theta \sim \mathcal{N}(0, I)$;
[2] *Apply random control signals and record data.*;
[3] **repeat**
[4] *Learn probabilistic (GP) dynamics model*;
[5] **repeat**
[6] *Approximate inference for policy evaluation*;
[7] *Gradient-based policy improvement*;
[8] *Update parameters* θ;
until *Convergence*;
[9] **return** θ^*;
[10] *Set* $\pi^* \leftarrow \pi(\theta^*)$;
[11] *Apply* π^* *to system and record data*;
until *Task Learned*;

Algorithm 7.1: PILCO algorithm

To find a policy π^* which converges to the desired target, PILCO builds a probabilistic GP dynamics model. This model provides the base for the deterministic approximate inference and policy evaluation, followed by the analytic computation of the policy gradients $\partial J^\pi(\theta)/\partial\theta$ for policy improvement. The policy π is improved based on the gradient information $\partial J^\pi(\theta)/\partial\theta$.

7.5 Simulation Setup

The underwater vehicle should be able to switch between different controllers in a single mission depending on the task to be solved at a specific time. The possible variants of controllers that an AUV can use in a mission are: follow bottom, depth control, follow pipe, go-to-point, path tracking, path following, etc. A series of different simulations were designed to evaluate the capability of PILCO to control an underactuated AUV within the three different evaluation scenarios: waypoint tracking, depth control and path tracking.

7.5.1 Vehicle Model

The vehicle model employed in this research is derived from semi-empirical calculation of the coefficients for the vehicle MULLAYA. MULLAYA is an underactuated AUV designed by the Defence Science and Technology Group as a research platform. The vehicle is controlled by a single propeller, a pair of elevator fins and a pair of rudder fins. General specifications of the vehicle are given in Table 7.1. The coefficients were calculated with the same technique as used by [66] and the obtained coefficients are presented in Table 7.2. As an engineering research platform, the vehicle undergoes multiple transformations over time in shape and internal engineering. This constant change leads to a need for constant updates of the controller for the vehicle. Possible future modification can include vectorised propulsion systems and buoyancy controllers.

Table 7.1: MULLAYA AUV particulars.

Property	**Value**	**Unit**
Length	1.56	**m**
Diameter	150	**mm**
Max RPM	3000	**RPM**
Weight	239.364	**N**
Buoyancy	246.2	**N**

Table 7.2: MULLAYA AUV coefficients employed in simulations.

Coeff.	**Result**	**Coeff.**	**Result**	**Coeff.**	**Result**
Xuu	-2.8	Zrp	0.68	Nuv	-30.88
Xwq	-29.6	Yuudr	12.12	Npq	-5.06
Xqq	-0.68	Nuudr	-7.51	Ixx	0.083
Xvr	29.6	Zuuds	-12.12	Iyy	3.08
Xrr	-4.95	Kpp	-1.30E-01	Izz	3.08
Yvv	-95.37	Kpdot	1.09E-02	Nwp	0.66
Yrr	-2.45	Mww	6.96	Nur	-5.31
Yuv	-32.9	Mqq	-135.04	Xudot	-0.51
Ywp	29.64	Mrp	5.1	Yvdot	-29.64
Yur	7	Muq	-5.34	Nvdot	0.66
Ypq	0.68	Muw	27.16	Mwdot	0.66
Zww	-95.37	Mwdot	-0.68	Mqdot	-5.05
Zqq	2.45	Mvp	-0.68	Zqdot	-4.94
Zuw	-32.9	Muuds	-7.738	Zwdot	-29.64
Zuq	-7	Nvv	6.96	Yrdot	4.95
Zvp	-29.64	Nrr	-135.03	Nrdot	5.05

7.5.2 Waypoint Tracking

The first type of controller evaluated for the underwater vehicle is a waypoint tracking controller. These types of controllers can be employed to transfer the vehicles between locations were more specific controllers are employed utilised or if it can be worked with the results of a path planning law that converts a desire path in waypoints. In this case, it is desiredthe desire was to control the vehicle velocity, azimuth and elevation such that the vehicle moves moved towards a specific location. The methodology used in this simulation is similar to the one employed in [151] but extended to 3D. If $\eta_{\mathbf{v}}$ is the vehicle state vector $[X_v, Y_v, Z_v, \phi_v, \theta_v, \psi_v]$ that can be divided in position vector $\mathbf{X_v}$ and orientation vector $\theta_{\mathbf{v}}$ on earth frame and the target point is expressed as the vector $\mathbf{X_T} = [X_d, Y_d, Z_d]$ the angles of the vector between the current position and the desired position can be expressed as

$$\begin{aligned} \psi_d &= \tan^{-1}\left(\frac{Y_d - Y_v}{X_d - X_v}\right) \\ \theta_d &= \tan^{-1}\left(\frac{\sqrt{(Y_d - Y_v)^2 + (X_d - X_v)^2}}{Z_d - Z_v}\right) \end{aligned} \tag{7.19}$$

and the vector of angle errors can be expressed as the difference between the vehicle orientation and the desire orientation, i.e $\theta_{\mathbf{e}} = [e_\psi, e_\theta, e_u]$, where $e_\psi = \psi_d - \psi_v$ and $e_\theta = \theta_d - \theta_v$. The target of the policy was to minimise the error angles to zero and the surge speed error to zero. In the proposed simulation a surge speed of $1.2m/s$ was set as the desired speed with an initial position of $[0, 0, 0]$,an orientation of $[0, 0, \pi/4]$ and an initial surge speed of $0.5m/s$. The first 12 seconds of real model simulation was employed to learn the policy with a target point with coordinates $[40, 40, 10]$. A constraint to limit the vehicle turn $< 180^o$ was applied to the process of policy testing over the vehicle model. A second simulation with two target points $[30, 30, 10]$ and $[90, 100, 10]$ was employed to check the viability of the policy. A total of 1000 sparse points was located as the limiter from which a sparse model was to be employed. The policy and simulation were executed at $5Hz$. A noise with variance of $\sigma^2 = 0.005$ was employed in the simulation and a variance of $\sigma^2 = 0.2$ was employed for the start position of the vehicle. When the vehicle arrived at 3 metres from the objective the vehicle was given the secondary target as the new objective. If the vehicle arrived at 1 meter of the final target the simulation would stop.

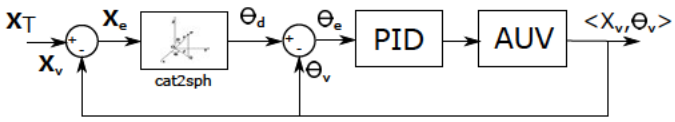

Figure 7.4: Control system block for waypoint tracking with PID controller.

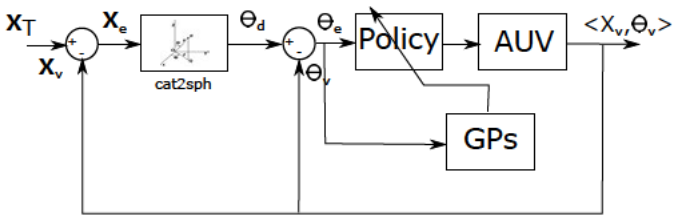

Figure 7.5: Control system blocks for waypoint tracking with RL policy.

7.5.3 Depth Control

A secondary scenario was explored to evaluate the controller's capability to keep a specific depth while travelling between location. This is an extension to the previous search of a policy, i.e. waypoint. In this scenario, the aim was for the vehicle to arrive at a specific depth before going to the desired location. The policy minimised the vector of error to zero with the difference that the vector of errors is $\theta_{\mathbf{e}} = [e_\psi, e_\theta, e_u, e_z]$, where $e_z = z_d - z_v$ similar to the waypoint tracking. The vehicle started from the same initial position of the waypoint tracking scenario $[0, 0, 0]$. The proposed training target was the vector $[40, 40, 5]$ and the selected test target points were $[30, 30, 5]$ and $[90, 100, 5]$. A total of 1500 sparse points was located as the limiter from which a sparse model would be employed. The policy and simulation were executed at $5Hz$. A noise of $\sigma^2 = 0.005$ was employed for the simulation and a $\sigma^2 = 0.2$ was employed for the start position of the vehicle. When the vehicle arrives at 3 metres of the objective the vehicle was given the secondary target as the new objective. If the vehicle arrived at 1 meter of the final target the simulation would stop.

7.5.4 Path tracking

With the aim of testing the capability of PILCO for path tracking, the scenario consisted of a single policy to control the propeller force, elevator force and rudder force of the

vehicle. The decision to learn the force and not to control the RPM and angle of the fins directly was to be able to compare the policy to a standard controller from the literature. With regards to the design of the path tracking simulation, the equations presented in section section 7.3 were employed. Figure 7.7 shows the block diagram of the LOS-PILCO control implementation whereby the policy would evolve based on the learned GPs. The target of all learned policies is to reduce the vector of errors $[e_\mu, e_v, e_\chi]$ to zero. An LOS-PID was used as a performance comparison. The LOS-PID (Figure 7.6)with the exact coefficient of the model was coded in the same way as [108] with the inclusion of measurement noise. The initial position of the vehicle was established as $[60, 3, 1]$ with an initial orientation of $[0, 0, 3\pi/4]$. A total of 1500 sparse points was located as the limiter from which a sparse model would be employed. A noise with $\sigma^2 = 0.001$ was employed throughout the simulation and a random variation of the start point with a variance of $\sigma^2 = 0.2$ was employed. Higher values of σ were not possible for the LOS-PID to be comparable as it was not able to overcome higher values of noise. Both the PID and PILCO policy were executed at a frequency of 10 Hz. A helix path (Figure 7.8) was parametrised as is shown in equation (7.20).

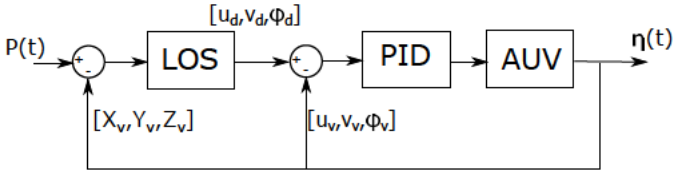

Figure 7.6: Control systems block for LOS-PID

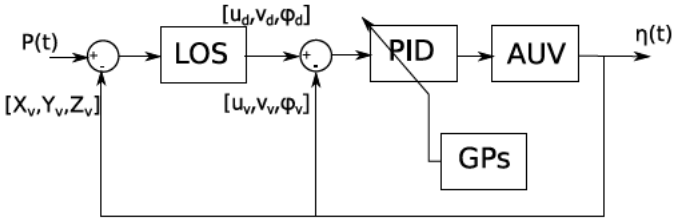

Figure 7.7: Control systems block for LOS-PILCO

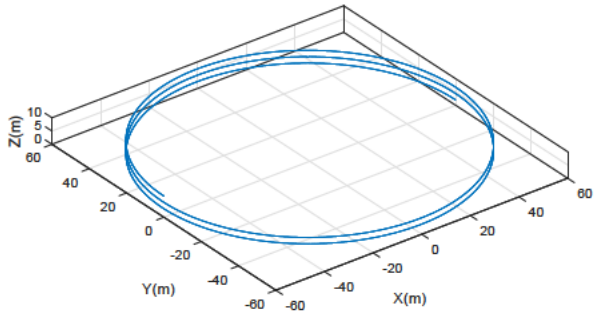

Figure 7.8: Spiral path to be followed

$$\begin{aligned} X_{helix} &= 60 * \cos(0.02618 * w_{ramp}(t)) \\ Y_{helix} &= 60 * \sin(0.02618 * w_{ramp}(t)) \\ Z_{helix} &= 2 + 2 * w_{ramp}(t)/200 \end{aligned} \tag{7.20}$$

where $w_{ramp}(t)$ is a function over time with a slope m. The control learning of the vehicle was carried out over the first 20 seconds at a frequency of 10 Hz.

7.6 Results

7.6.1 Waypoint Tracking

As evident in Figure 7.11, the learning of a policy for waypoint tracking of an AUV is possible with the application of PILCO. However, reinforced learning does not require the calibration of parameters but, rather requires the tuning of the parameters of the **Q** matrix from the cost function. For the cost function design, the surge speed was given a higher importance in the cost function than the other error vectors. This escalation of each target in the learning policy was needed as an underactuated AUV needs to get to a specific speed to be able to dive and navigate with a more linear model. An example of the data employed to create the GPs model can be seen in Figure 7.9.

These simulations showed that the required number of episodes over the platform was under 25 to obtain a usable policy to control the vehicle and be able to arrive at all three targets. Figure 7.10 shows the evolution of the cost function over time for the platform.

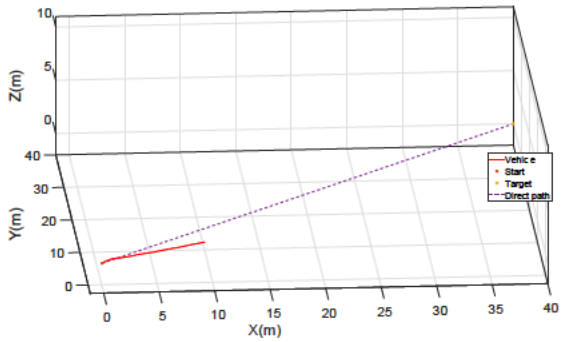

Figure 7.9: Real model waypoint tracking data example for GPs learning.

The cost function drops rapidly to a low value as the vehicle learns to control the surge speed and from there, how to control its orientation. The decision to do this was based on the fact that the control surfaces such as fins require a minimum speed to be useful.

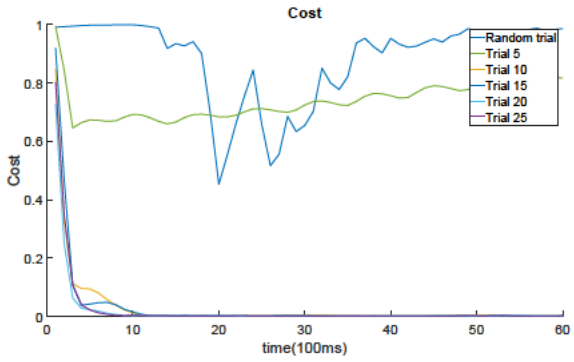

Figure 7.10: Waypoint tracking cost function evolution.

The result from the final policy selected is shown in Figure 7.11; plotting against the results from the PID controller for the same scenario. Both PID and PILCO arrive near to both test points. However, PILCO shows a more direct route taken towards the targets.The PID controller took 1201 cycles to arrive, equivalent to 240.2 seconds, and the PILCO policy took 569 cycles equivalent to 113.8 seconds. PILCO has an advantage over a simple PID as PILCO has learned the policy over a noisy platform(or environment) and the PID does not have any component to compensate for the noise in the measurement. For PID to compensate for the noise an observer will have to be designed and implemented. However, this will increase the complexity of the controller and the need for more design and calibration time for the PID controller. The results

show that the waypoint tracking with a PILCO controller can be a viable option as the tuning of the controller coefficients are practically zero. The robustness of the PILCO policy is higher as the platform will start the learning from different angles and can overcome noise in the measurement. The policy is also updated in case of failure. A down point of PILCO in the simulated scenarios is that the controller tries to always have the same behaviour, e.g. if the vehicle dives a little before directing to the first point after the second point the policy will try to repeat a similar behaviour.

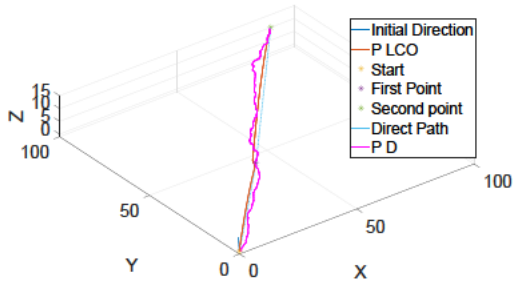

Figure 7.11: 3D view waypoint tracking comparative result between PILCO and PID results $[m]$.

7.6.2 Depth Control

The simulation of simultaneous waypoint tracking, and depth control of underwater vehicles shows that PILCO can learn a complex policy to control the vehicle. An example of the training data used for learning the vehicle GPs model can be seen in Figure 7.12. The quality of the learning of the GPs model is directly related to the quality of the learning of the policy. After 20 episodes over the platform a good quality GPs model was learned such that a policy could be learned. The evolution of the cost function over time can be seen in Figure 7.13. In similar way to the standard waypoint tracking, the cost evolves very quickly to a low value and then small changes are carried out in the policy learning. This small change is produced by the scaling of each of the targets to be obtained.

Figure 7.14 shows the results of policy 22. The policy is capable of keeping the depth at near to 5 metres and at the same time keep all four errors at near zero value. Usu-

ally, the task described here will be conducted with two controllers: a depth controller and an azimuth controller. A single controller with two targets(elevation, depth) was applied that at the start go on different paths. The $[X, Y, Z]$ results from the simulation are presented in Figure 7.15. The policy produces a different behaviour than normal controllers, if by operator error a target was setup with a different depth than the desire depth. The learned policy will try to push the vehicle to the target disregarding the depth by a small quantity such that the target can be completed, and this can be observed in Figure 7.16 . Three targets were set up as $[30, 30, 5]$, $[70, 70, 3]$, and $[90, 100, 5]$ the second target being outside of the desired depth of the policy.

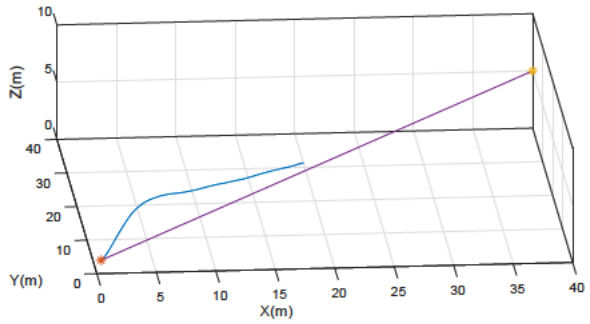

Figure 7.12: Real model waypoint tracking and depth control data example for GPs learning.

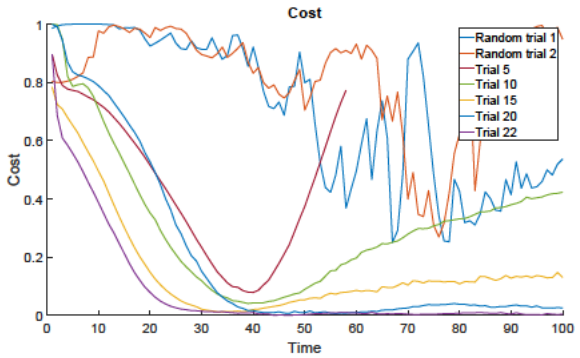

Figure 7.13: Waypoint tracking and depth control cost function evolution.

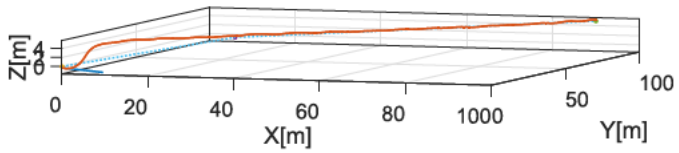

Figure 7.14: 3D view waypoint tracking and depth control PILCO results.

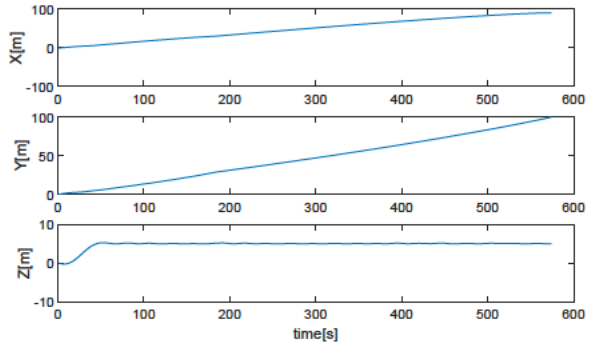

Figure 7.15: $[X, Y, Z]$ waypoint tracking and depth control PILCO results.

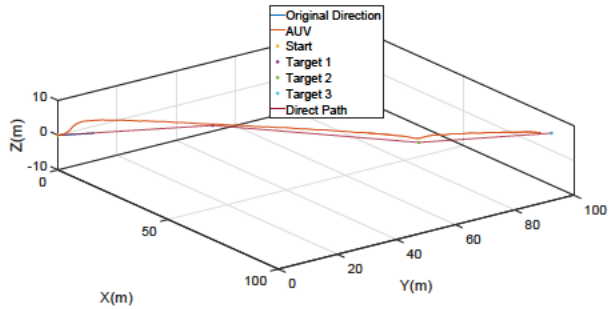

Figure 7.16: 3D waypoint tracking and depth control PILCO results with a target outside of the control depth.

7.6.3 Path Tracking

The training evolution of the GPs model in the direction of a better policy allows PILCO to search for a better policy after each iteration with the real model (Figure 7.18). Figure 7.17 presents an example of the data used for training the GPs model that was later used for policy search. The reinforced learning simulation to learn an LOS controller with PILCO shows that the reinforced learning algorithm can learn to follow the desired path based on the desired angles produced by an LOS law. The policy can perform better than an LOS-PID controller that is without noise compensation algorithms. The results from the PID to follow the LOS law can be seen in Figure 7.19, without noise the PID can stabilise itself very quickly but with minimum noise in the depth sensor the χ angle cannot be completely stable and the controller will fluctuate. In the same way,the PILCO controller (Figure 7.20) reacts to the noise of the measurement but can follow the desired path in a better way than the LOS-PID controller implemented.

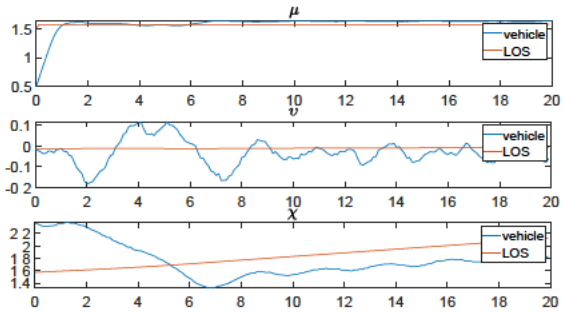

Figure 7.17: Sample of data used for learning of GPs model employed.

The measurement of the RMSE between the desired vehicle position and the vehicle position for both LOS-PID and LOS-PILCO is presented in Table 7.3. LOS-PILCO is shown to out-perform the PID controller in the reduction of error between the LOS law and the vehicle position. Not only in the accuracy of placing the vehicle in the correct position but in the speed of deployment of a controller to follow LOS whereas LOS-PILCO will require less tuning with field tests. Another advantage of PILCO is the absence of knowledge from the vehicle coefficients and not need to use other vehicles variables such as surge, heave and sway speeds.

Figure 7.21 and Figure 7.22 present the comparative plots of position in X, Y, Z and the

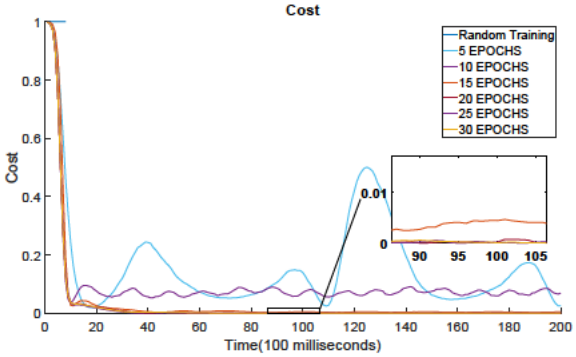

Figure 7.18: Cost evolution for each policy test over real model.

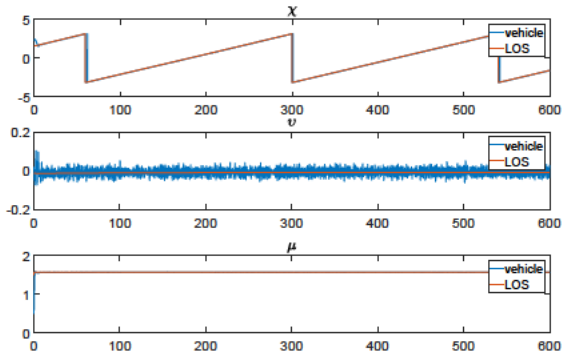

Figure 7.19: Desire LOS angles and speed and vehicle LOS angles and speed produced by PID controller.

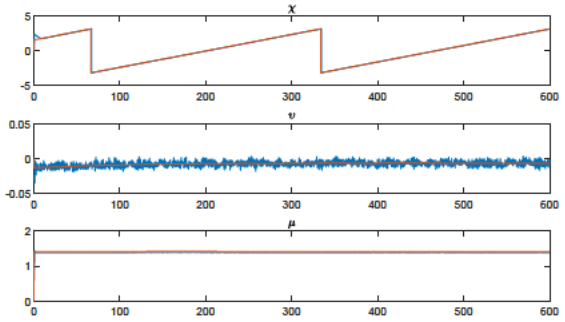

Figure 7.20: Desire LOS angles and speed and vehicle LOS angles and speed produced by PILCO policy.

Table 7.3: RMSE results from LOS-PID and LOS-PILCO.

RMSE	Value
LOS-PID	1.214
LOS-PILCO	0.89

3D path taken by the vehicle with PID controller and PILCO policy. Both controllers shows their ability to direct the AUV to the desired path, most of the policies after the episode 20 show better performance than a LOS-PID.

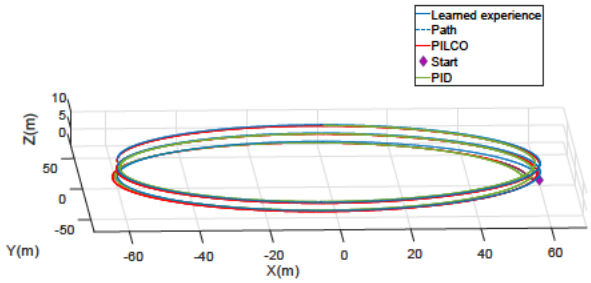

Figure 7.21: 3D comparison of vehicle controlled with PID and PILCO controller.

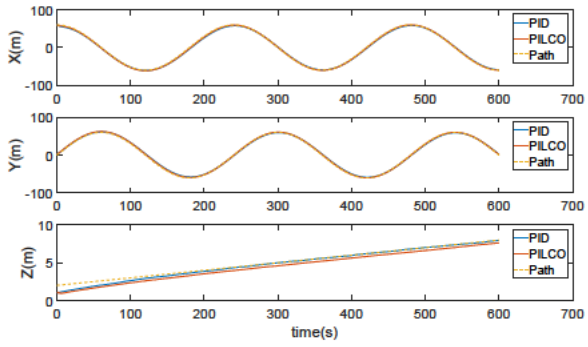

Figure 7.22: X,Y,Z comparison of vehicle controlled with PID and PILCO policy.

7.7 Conclusions

This chapter has investigated the applicability of PILCO algorithms and its requirements to learn policies to control underactuated AUVs. Three sets of simulations were designed

to test different applications such as waypoint tracking, depth control and path tracking. A simple waypoint tracking control can be learned in a small quantity of experiments over the real platform, and the performance of the obtained policy was compared with a PID controller which was over-performed by the policy as the policy obtained was learned over the platform including noise.

In a similar way, a depth control policy was learned by mixing the waypoint tracking objective with the depth objective. However, in this research the learning of a policy to only control depth was not successful as the GPs model to learn the policy requires more information to allow the learning of the policy and there is a coupling between the rudder and elevator as the vehicle rolls 5 degrees over its axis. Nevertheless, in the simulation of depth control it had been shown that a policy of depth control can be obtained by the learning of simultaneous waypoint tracking and depth control. The combination of objectives yields a more intuitive behaviour, similar to what a human will do with the proposed objectives.

In the case of the proposed PILCO-LOS methodology, it has been shown that PILCO is a viable option to learn a policy to minimise the error between an LOS law and the vehicle position. The comparison with the previously formulated PID controller shows that the learned policy will perform equally and sometimes better than a formulated PID. The RMSE shows that PILCO can obtain better performance and the long period of simulation shows that the learned policy can constantly minimise the error to the desired value.

The PILCO algorithm has shown that it is applicable to an underactuated AUV. The vehicle shape and actuators will force the selection of values of the cost function. However, reinforced learning for an underactuated AUV has been shown to be a solution to the control of underwater vehicles and can be a solution to control bioinspired vehicles as their mathematical model is more complex.

Chapter 8

Summary, Conclusions & Future work

This chapter brings together the findings of the individual chapters. It also concludes the findings and outcomes, and discusses the implications of the findings, the limitations, and the recommendations for further research.

8.1 Summary of Work Performed

In this thesis, the application of GPs (Gaussian processes) for SI (system identification), inertial navigation, and control of underwater vehicles were examined. This thesis is an effort to answer the question, "Can GPs be used for system identification of AUVs and what applications can benefit from the use of a non-parametric model?" As the first step in addressing this question, it was required to first develop a methodology for underatuated ships. Once a suitable methodology was identified, the next step was to extend the methodology to underatuated AUVs. After probing that GPs can be applied to SI of AUVs a series of application were explored. GPs are a strong element of machine learning with a large field of possible application to underwater vehicles. The applications explored in this thesis are related directly to the present and future development of the platform MULLAYA AUV.

In the first step, a simpler model of a container ship was chosen to verify that GPs are able to identify a marine vehicle. A GPs composed of a multi-output kernel and a vector of inputs were chosen to maintain the relation between the outputs and inputs of the system as the chosen ship model has high coupling between its inputs and outputs. The simulation to test the viability were done over the available model of a container ship. The root mean square error (RMSE) and predicted residual error sum of squares (PRESS) were employed to measure the difference between the proposed model and a neural network system identification.

The results of the first step provided a build-up approach towards extending GPs to the SI of underactuated AUVs (Chapter 3), given that the latter has a higher degree-of-freedom (DoF) and coupled dynamics. The required time variable, the number of regressors, and type of data required to model underwater vehicles with multi-output GPs were identified. A series of simulations that employed the model of a Remus 100 were designed to test the robustness of the model. The RMSE and PRESS were employed to measure the difference between the proposed model and a neural network system identification.

The positive results of SI of underactuated AUVs allows the exploration of application that may help the development of MULLAYA AUV. The first application explored

(Chapter 4) examines the ability of non-parametric models to be used as the transfer function for the post-processing of pressure sensor data for speed measurement. The vehicle MULLAYA had in the past been used as a research platform to measure speed with pressure sensors located in the vehicle head. The original idea was to use a single differential pressure sensor connected between the main point of the nose and the lateral points of measurement. However, the research was never completed. This application employed a series of experiments carried out with MULLAYA to create a speed measurement unit via the use of an array of pressure sensors. This configuration was different from the original design, as multiple pressure points are measured and post-processing techniques are employed to calculate the speed. Three techniques were analysed: a standard parametric technique, neural network post-processing, and GPs post-processing. The comparison between all three techniques allows the analysis of the performance of each technique under non-linearities.

The second application explored GPs for the prediction and correction of the vehicle states in a navigation system (Chapter 5). This application was selected as MULLAYA AUV does not possess a commercial navigation system but rather has a simple inertial navigation unit and a compass. A GPs non-parametric model from an underwater vehicle was employed as the internal model of an unscented Kalman filter (UKF). This technique call GP-UKF was compared with an ideal UKF and multiples measurement of error were taken to compare the ability to correct and predict each component of the vehicle state.

For the final application explored, the calculation of the hydrodynamic coefficients by semi-empirical methodologies for the platform MULLAYA AUV was required (Chapter 6), a script that generates the required coefficients and the results of a series of experiments with the reconditioned was employed for calculation and validation.

Finally, with the calculated mathematical model, the application of model-based reinforced learning with a GPs non-parametric model has been explored via the application of the Probabilistic Inference for Learning Control (PILCO)(in Chapter 7). Three different controllers were tested: waypoint tracking, depth control, and LOS (line-of-sight). For each variation of control, a comparative controller was implemented and measure-

ments of effectiveness were taken. This comparative study allows the analysis of the performance of the obtained policies.

8.2 Findings

The major findings of the research are listed below.

8.2.1 System Identification

- Multi-output GPs with NARX (Nonlinear autoregressive exogenous model) structure is an effective technique for producing non-parametric model for underactuated marine vehicles, as preserves the relation between the outputs.
- SI of underactuated AUVs with GPs requires the normalization of the inputs vector but not from the outputs vector in the learning stage
- Single output GPs may be able to identify an AUV. However, the performance outside of the learning horizon is higher for a multi-output GPs.
- SI with NARX structure from underactuated marine vehicles requires at least two regressors.
- GPs perform better that standard neural network methodologies outside of the learning horizon.
- Sparse GPs techniques reduce computational cost of learning and deployment of GPs.

8.2.2 Non-Parametric Model Application to AUV

- A non parametric model from a GPs can be employed as a non-linear transfer function for prediction of vehicle speed based in an array of pressure sensors.
- Pressure sensors is an effective technique for measurement of vehicle speed if the non-linearity of the vehicle produced by the acceleration are included.
- The acceleration of the vehicle generates non-linearity in the pressure profiles around an underwater vehicle.
- The inertial navigation of an underwater vehicle can be done effectively with GP-

UKF, the inclusion of the non-parametric model allows a faster implementation of the UKF algorithm and the resultant system is capable of predict the vehicle position as well as an ideal parametric model.

- A small quantity of real experiments is required to learn a motion control policy for underwater vehicles if model-based reinforced learning is applied.
- The evolution of the non-parametric model by experimentation of the policy learned through reinforced learning is an effective technique to reduce the number of experiments require.
- The cost function for policy learning needs to give priority to reduce the speed of the vehicle before minimising the other objectives as this makes the policy converge faster to the objective.

8.3 Conclusions

In this study, it was shown that multi-output GPs and single output GPs can model marine vehicles behavior. In the case of multi-output GPs, the learned model outperforms outside of the learned horizon other techniques such as neural networks. The results mean that multi-output GPs can be used for techniques such as control model prediction. A specific case where GPs technique can be employed is when the vehicle needs to navigate backwards with high precision. Standard techniques of modeling will require a completely different mathematical model than the original, which is usually designed only for a forward movement. In the case of GPs, a single model can be learned by pairing the information of the vehicle in both forward and backward motions during the learning process and techniques such as sparse GPs can further be used to reduce the computational cost.

In the case of underactuated AUVs, the need for techniques such as multi-output GPs is important as each DoF is coupled to each other. As the vehicle does not advance with a zero roll, the control surfaces produce a small force and moment in the adjacent DoF. In the same way, the absence of direct control surfaces for some DoF requires the vehicle to have a small pitch rotation such that the vehicle's nose points to a specific angle. Multi-output GPs have shown their viability to represent underwater vehicles

inside and outside of the learned horizon.

In the application explored on this study, non-parametric based techniques performed better than the common parametric technique as they do not only consider the linear model but also the non-linear modelling. GPs model was found to have a notable advantage in that it can learn to overcome noisy measurements. All three explored applications are highly affected by environmental noise (e.g. currents, waves, magnetic noise). A standard mathematical model to replicate what GPs are capable of doing, it will need to be able to overcome the noise by the addition of special models in order to predict environmental variables.

The application with possible the highest impact is the application of reinforced learning for underwater vehicle based on the use of non-parametric models. All explored scenarios showed the ability to obtain a policy capable of completing the defined task with a few episodes, usually less than 50 episodes for the autonomous platform compared to the number of episodes that would be required with a technique such as deterministic policy gradient. The advantage of PILCO is clearly appreciated in the application of control. The learning over noisy signal gives an advantage over standard controllers as the policy is fault tolerant against the signal noise.

GPs applied to SI and its applications show advantages over parametric methodologies for underwater vehicles. The presence of the variance from the GPs is a strong tool of the confidence measure that showed it can be exploited in probabilistic algorithms as a Kalman filter. The ability of GPs to model in a more precise and robust form of natural phenomenon is an advantage that can be seen in the results as GPs are able to model an AUV over a noisy environment. The Bayesian nature of the noise that affects AUVs allows their representation by the internal kernels of GPs. The non-parametric representation with higher fidelity allows a better modelling of the vehicle with the added benefit that for an AUV, the non-parametric can be obtained more quickly than with a mathematical model. GPs can become an important tool for AUVs as they allow faster learning on transfer functions and vehicle models that can be used for different tasks, as explored in this thesis.

8.4 Future Work

The findings of this thesis could lead to several further investigations into the applicability of GPs to underwater vehicles for direct application similar to those explored in this thesis or hybrid applications where GPs are employed with other techniques. There are areas that were treat on this study that could not be fully covered due to time and scope limitations of this project.

In particular, more real field trials are required to design a speedometer based on pressure sensors with GPs post-processing. Real field experiments are required where the real velocity can be logged with DVLs(Doppler velocity logger) and the pressure from the vehicle nose is measured. Another avenue of possible future research concerns the measurement of currents with ADCP(acoustic Doppler current profiler) from underwater vehicles having problems with removing the vehicle relative speed from the real current velocity. If both the ADCP signal and the pressure signal are post-processed through a GPs, this can result in the measurement of the absolute velocity of the current.

The Multi-output GPs learned in Chapter 2 and Chapter 3 can be cascaded to be employed as the internal model for model predictive control for motion control of underwater vehicles. Similarly, multi-output GPs can be employed as an observer to overcome noise signals in the same way that a neural network is employed. Multi-output GPs can be also be used as the internal model of GP-UKF to improve the results found in this research. In addition, the multi-output GPs model can be also employed as the internal model of PILCO RL.

PILCO application needs to be further explored as there are multiple applications that were not tested in this research. Tasks such as following pipe, diving from the surface and keeping position near to the surface can take advantage of the application of PILCO. Another issue to explore through research is the application of GPs to bio-inspired vehicles: bio-inspired vehicles can be fitted with GPs for both navigation and control as their mathematical models are complex.

Additionally, there are new machine learning algorithms for reinforced learning that improve PILCO as an example with integrating Bayesian neural network dynamics models,

also called DeepPilco. These algorithms can be researched to determine whether there is any benefit to their application in underwater vehicles.

Appendix A

REMUS 100 Matlab Model

```
function xdot= REMUS_KFILTER(x,ui)
% Simulation algorithm for REMUS This code is based in:
% Prestero, T., 2001. Verification of a six-degree of freedom simulation
% model for the REMUS autonomous underwater vehicle (Doctoral dissertation,
% Massachusetts Institute of Technology and Woods Hole Oceanographic
% Institution).

%                 -----F1-----------------------------------
%           |   /             ||                            ||    \
%           |==       F3      ||         Remus              ||     |
%           |   \-------------||-------------------------||---/
%                             F2
% V=[u,v,w,p,q,r,x,y,z,phi,psi,theta];


% TERMS
% -------------------------------------------------
%STATE VECTOR:
%
% x = (u v w p q r xpos ypos zpos phi theta psi) ,
% Body-referenced Coordinates
% u  = Surge velocity [m/s]
% v  = Sway velocity  [m/s]
% w  = Heave velocity [m/s]
% p  = Roll rate      [rad/s]
% q  = Pitch rate     [rad/s]
% r  = Yaw rate       [rad/s]
%
% Earth-fixed coordinates
% xpos  = Position in x-direction [m]
% ypos  = Position in y-direction [m]
% zpos  = Position in z-direction [m]
% phi   = Roll angle              [rad]
% theta = Pitch angle             [rad]
% psi   = Yaw angle               [rad]
%
%INPUT VECTOR
% ui = [n delta_s delta_r]'
%  Control Fin Angles
%   n= shaft speed RPM
%   delta_s = angle of stern planes  [rad]
%   delta_r = angle of rudder planes [rad]

% Get state variables
u     = x(1,:);
v     = x(2,:);
w     = x(3,:);
p     = x(4,:);
q     = x(5,:);
r     = x(6,:);
phi   = x(10,:);
theta = x(11,:);
psi   = x(12,:);

% Get control inputs
delta_s = -ui(2,:);
delta_r = ui(3,:);
```

```
% 45 degree maximum rudder angle
delta_max = 45 * pi/180;

% Check control inputs (useful later)
% if abs(delta_s) > delta_max && delta_s < delta_max
%     delta_s = sign(delta_s) * delta_max;
% end
%
% if abs(delta_r) > delta_max
%     delta_r = sign(delta_r) * delta_max;
% end

% Initialize elements of coordinate system transform matrix
% ------------------------------------------
    ------------------------------------------
c1 = cos(phi);
c2 = cos(theta);
c3 = cos(psi);
s1 = sin(phi);
s2 = sin(theta);
s3 = sin(psi);
t2 = tan(theta);

% ------------------------------------------
% Vehicle Parameters and Coefficients
% ------------------------------------------
W = 2.99e2; % Weight (N)
B = 2.99e2; % Bouyancy (N)%% Note buoyanci incorrect simulation fail with
    this value

g = 9.81; % Force of gravity
m = W/g; % Mass of vehicle

Xuu   = -1.62;    % Axial Drag
Xwq   = -3.55e1;  % Added mass cross-term
Xqq   = -1.93;    % Added mass cross-term
Xvr   =  3.55e1;  % Added mass cross-term
Xrr   = -1.93;    % Added mass cross-term
Yvv   = -1.31e3;  % Cross-flow drag
Yrr   =  6.32e-1; % Cross-flow drag
Yuv   = -2.86e1;  % Body lift force and fin lift
Ywp   =  3.55e1;  % Added mass cross-term
Yur   =  5.22;    % Added mass cross-term and fin lift
Ypq   =  1.93;    % Added mass cross-term
Zww   = -1.31e2;  % Cross-flow drag
Zqq   = -6.32e-1; % Cross-flow drag
Zuw   = -2.86e1;  % Body lift force and fin lift
Zuq   = -5.22;    % Added mass cross-term and fin lift
Zvp   = -3.55e1;  % Added mass cross-term
Zrp   =  1.93;    % Added mass cross-term

% Center of Gravity wrt Origin at CB
xg = 0;
yg = 0;
zg = 1.96e-2;

% Control Fin Coefficients
Yuudr =  9.64;
```

```
Nuudr = -6.15;
Zuuds = -9.64; % Fin Lift Force

% Center of Buoyancy wrt Origin at Vehicle Nose
xb = 0;%-6.11e-1;
yb =  0;
zb =  0;
n=ui(1,:)/60*2*pi;
% Propeller Terms
Xprop =  1.569759e-4*n.*abs(n);
Kpp   = -1.3e-1;  % Rolling resistance
Kprop = -2.242e-05*n.*abs(n);%-5.43e-1; % Propeller Torque
Kpdot = -7.04e-2; % Added mass

% Cross flow drag and added mass terms
Mww   =  3.18; % Cross-flow drag
Mqq   = -1.88e2; % Cross-flow drag
Mrp   =  4.86; % Added mass cross-term
Muq   = -2; % Added mass cross term and fin lift
Muw   =  2.40e1; % Body and fin lift and munk moment
Mwdot = -1.93; % Added mass
Mvp   = -1.93; % Added mass cross term
Muuds = -6.15; % Fin lift moment
Nvv   = -3.18; % Cross-flow drag
Nrr   = -9.40e1; % Cross-flow drag
Nuv   = -2.40e1; % Body and fin lift and munk moment
Npq   = -4.86; % Added mass cross-term

% Moments of Inertia wrt Origin at CB
Ixx  = 1.77e-1;
Iyy  = 3.45;
Izz  = 3.45;

Nwp = -1.93; % Added mass cross-term
Nur = -2.00; % Added mass cross term and fin lift

% Non-linear Moments Coefficients
Xudot = -9.30e-1; % Added mass
Yvdot = -3.55e1;  % Added mass
Nvdot =  1.93;    % Added mass
Mwdot = -1.93;    % Added mass
Mqdot = -4.88;    % Added mass
Zqdot = -1.93;    % Added mass
Zwdot = -3.55e1;  % Added mass
Yrdot =  1.93;    % Added mass
Nrdot = -4.88;    % Added mass

% Set total forces from equations of motion
% ------------------------------------------------------------
X = -(W-B)*sin(theta) + Xuu*u.*abs(u) + (Xwq-m)*w.*q + (Xqq + m*xg)*q.^2
    ...
    + (Xvr+m)*v.*r + (Xrr + m*xg)*r.^2 -m*yg*p.*q - m*zg*p.*r  ...
    + Xprop  ;

Y = (W-B)*cos(theta).*sin(phi) + Yvv*v.*abs(v) + Yrr*r.*abs(r) + Yuv*u.*v
    ...
    + (Ywp+m)*w.*p + (Yur-m)*u.*r - (m*zg)*q.*r + (Ypq - m*xg)*p.*q  ...
```

```
    + Yuudr*u.^2.*delta_r ;

Z = (W-B)*cos(theta).*cos(phi) + Zww*w.*abs(w) + Zqq*q.*abs(q)+ Zuw*u.*w
    ...
    + (Zuq+m)*u.*q + (Zvp-m)*v.*p + (m*zg)*p.^2 + (m*zg)*q.^2 ...
    + (Zrp - m*xg)*r.*p + Zuuds*u.^2.*delta_s ;

K = -(yg*W-yb*B)*cos(theta).*cos(phi) - (zg*W-zb*B)*cos(theta).*sin(phi)
    ...
    + Kpp*p.*abs(p) - (Izz- Iyy)*q.*r - (m*zg)*w.*p + (m*zg)*u.*r + Kprop ;

M = -(zg*W-zb*B)*sin(theta) - (xg*W-xb*B)*cos(theta).*cos(phi) + Mww*w.*abs
    (w) ...
    + Mqq*q.*abs(q) + (Mrp - (Ixx-Izz))*r.*p + (m*zg)*v.*r - (m*zg)*w.*q
    ...
    + (Muq - m*xg)*u.*q + Muw*u.*w + (Mvp + m*xg)*v.*p ...
    + Muuds*u.^2.*delta_s ;

N = -(xg*W-xb*B)*cos(theta).*sin(phi) - (yg*W-yb*B)*sin(theta) ...
    + Nvv*v.*abs(v) + Nrr*r.*abs(r) + Nuv*u.*v ...
    + (Npq - (Iyy- Ixx))*p.*q + (Nwp - m*xg)*w.*p + (Nur + m*xg)*u.*r ...
    + Nuudr*u.^2.*delta_r ;


FORCES = [X Y Z K M N]';

% Accelerations Matrix (Prestero Thesis page 46)
Amat = [(m - Xudot) 0                0                0              m*zg
          -m*yg;
        0             (m - Yvdot)    0                -m*zg          0
          (m*xg - Yrdot);
        0             0              (m - Zwdot)      m*yg           (-m*xg -
    Zqdot) 0;
        0             -m*zg          m*yg             (Ixx - Kpdot) 0
          0;
       m*zg           0              (-m*xg - Mwdot) 0              (Iyy -
    Mqdot)    0;
        -m*yg         (m*xg - Nvdot) 0                0              0
          (Izz - Nrdot)];

% Inverse Mass Matrix
Minv = inv(Amat);

% Derivatives
xdot = ...
    [Minv(1,1)*X + Minv(1,2)*Y + Minv(1,3)*Z + Minv(1,4)*K + Minv(1,5)*M +
    Minv(1,6)*N;
     Minv(2,1)*X + Minv(2,2)*Y + Minv(2,3)*Z + Minv(2,4)*K + Minv(2,5)*M +
    Minv(2,6)*N;
     Minv(3,1)*X + Minv(3,2)*Y + Minv(3,3)*Z + Minv(3,4)*K + Minv(3,5)*M +
    Minv(3,6)*N;
     Minv(4,1)*X + Minv(4,2)*Y + Minv(4,3)*Z + Minv(4,4)*K + Minv(4,5)*M +
    Minv(4,6)*N;
     Minv(5,1)*X + Minv(5,2)*Y + Minv(5,3)*Z + Minv(5,4)*K + Minv(5,5)*M +
    Minv(5,6)*N;
     Minv(6,1)*X + Minv(6,2)*Y + Minv(6,3)*Z + Minv(6,4)*K + Minv(6,5)*M +
    Minv(6,6)*N;
     c3.*c2.*u + (c3.*s2.*s1-s3.*c1).*v + (s3.*s1+c3.*c1.*s2).*w;
```

```
  s3.*c2.*u + (c1.*c3+s1.*s2.*s3).*v + (c1.*s2.*s3-c3.*s1).*w;
-s2.*u+c2.*s1.*v+c1.*c2.*w;
  p+s1.*t2.*q+c1.*t2.*r;
  c1.*q-s1.*r;
  s1./c2.*q+c1./c2.*r] ;
```

Appendix B

GP-UKF for AUV Matlab Implementation

```
clc;clear all;
% some defaults for the plots
set(0,'defaultaxesfontsize',30);
set(0,'defaultaxesfontunits', 'points')
set(0,'defaulttextfontsize',33);
set(0,'defaulttextfontunits','points')
set(0,'defaultaxeslinewidth',0.1);
set(0,'defaultlinelinewidth',2);
set(0,'DefaultAxesLineStyleOrder','-|--|:|-.');

% Parameters for UKF
alpha = 1; beta  = 0; kappa = 0;


afun=@fAUVRemus;

%% Learn Models
% Load data
load X.mat
load Y.mat
load U.mat
Real_data=X(2:end,:)';
time_data=X(1,:)';
X=resample(X',1,1)';
time=X(1,1:end)';
Y=resample(Y',1,1)';
U=resample(U',1,1)';
[~,steps]=size(X);


% covariance function
covfunc={'covSum',{'covSEard','covNoise'}};

% learn dynamics model
% determine how many elements is 50%
numelements = round(0.4*steps);
% get the randomly-selected indices
s = rng;
sprev = rng('shuffle','twister');
indices = randperm(steps);
indices = sort(indices(1:numelements));


yd=X(2:end,indices)';
%B=[7:12  1:6];
%yd(:,[1:12]) = yd(:,B);
[m,n]=size(yd);
yd_delay=[zeros(1,n);yd(1:end-1,:)];
xd=[U(2:end,indices)' yd_delay];
tic;
Xd = trainf(xd,yd);  disp(exp(Xd))
toc


% learn observation model
xm = yd;
ym = Y(2:end,indices)';
tic;
```

```
Xm = trainf(xm,ym);   disp(exp(Xm))
toc



%% some error measures
sfun = @(xt, x, C) (x - xt).^2; % squared distance
smfun = @(xt, x, C) (x - xt).^2./C./length(x); % squared Mahalanobis
    distance per point
mfun = @(xt, x, C) sqrt((x-xt).^2./C)./length(x); % Mahalanobis distance
    per point
nllfun = @(xt, x, C) (0.5*log(C) + 0.5*(x-xt).^2./C + 0.5.*log(2*pi))./
    length(x); % NLL per point


%% State estimation
T = 2;          % length of prediction horizon
noTest = steps; % size of test set
u = U(2:end,1:steps)'; % means of initial states
%y = zeros(1, noTest);  % observations

% Considered estimators: (1) ground truth, (2) ukf, (3) gpf, (4) ekf
xp = zeros(noTest,T+1); % predicted state
xe = zeros(12,noTest,T+1); % filtered state
xy = zeros(noTest,T+1); % predicted measurement (mean)
Cy = zeros(noTest,T+1); % predicted measurement (variance)
Cp = zeros(noTest,T+1); % predicted variance
Pest=pascal(12);
Pest=Pest/(10*Pest(12,12));
Ce=cell(noTest,T+1);
[Ce{:}]=deal(Pest);
%Ce = 0.5*ones(noTest,T+1); % filtered variance
y=Y(2:end,1:steps)';
% xe(:,:,1)= x';
% xe(1,:,1) = chol(C)'*randn(noTest,1) + x';
%Ce(:,:,1) = repmat(C,5,noTest);
tic
for t = 1:T

  for i = 1:steps-1

   %------------------------------------ GP-UKF
    ------------------------------------

     [xEst,PEst,xPred,PPred,zPred,S,K] = ...
       gpukf(xe(:,i,t),cell2mat(Ce(i,t)),u(i,:), Xd, xd, yd, y(i,:)', Xm, xm
    , ym, alpha, beta, kappa);
  xe(:,i,t+1)=xEst;
  xe(:,i+1,t)=xEst;
  Ce{i,t+1}=PEst;
  Ce{i+1,t}=PEst;

  end

end
toc
%% Plot
Aux=xe(:,:,1)';
%t=linspace(0,steps*0.2,steps);
```

```
names=["u","v","w","p","q","r","X","Y","Z","roll","pitch","yaw"];
for i=1:12
    figure(i);clf;
    plot(time_data,Real_data(:,i),'b');hold on
    plot(time,Aux(1:steps,i),'r');
    title(names(i))
    legend('Original Data','GP-UKF')
    saveas(gcf,strcat(names(i),'-UKF'),'meta')
    saveas(gcf,strcat(names(i),'-UKF','.fig'))
end
%%
figure(16);clf;
X=X(2:end,:)';
plot3(X(:,7),X(:,8),-X(:,9),'b');hold on
plot3(X(indices,7),X(indices,8),-X(indices,9),'-r');
plot3(Aux(1:steps,7),Aux(1:steps,8),-Aux(1:steps,9),'g');
plot3(X(1,7),X(1,8),-X(1,9),'*r');hold off;
az = 180+45;
el = 45;
view(az, el);
saveas(gcf,strcat('plot3','-UKF'),'meta')
saveas(gcf,strcat('plot3','-UKF','.fig'))
%X=X(2:end,:)';
RMSE = sqrt(mean((X - Aux).^2));  % Root Mean Squared Error
clc;
Y = sprintf('The RMSE of the system is  ');
disp(Y);
disp(RMSE);
filename='REMUS_GP_UKF_RESULT.mat';
REMUS_GP_UKF_RESULT=Aux;
save(filename,'REMUS_GP_UKF_RESULT');
poolobj = gcp('nocreate');
delete(poolobj);
%%% UKF section
%%load Ynoise.mat;
%%ym = Ynoise(2:end,1:steps)';
%y=ym;
%afun=@fAUVRemus;
%hfun=@HAUVRemus;
%   Cw = [0.9 0.01 0.01 0.1 0.1 0.1 0.2 0.2 0.2 0.2 0.2 0.2 ];
%   Cv = [0.2 0.2 0.2 0.2 0.2 0.2 0.3 0.3 0.3 0.3 ];
%Pest=pascal(12);
%Pest=Pest/(100*Pest(12,12));
%Ce=cell(noTest,T+1);
%[Ce{:}]=deal(Pest);
%for t = 1:T
%   for i = 1:steps
%
%     %------------------------------------ UKF
    ------------------------------
%      [xEst,PEst,xPred,PPred,zPred,S,K] = ...
%        ukf_add(xe(:,i,t),cell2mat(Ce(i,t)),u(i,:), Cw, afun, y(i,:)', Cv,
    hfun, 1, alpha, beta, kappa);
%   xe(:,i,t+1)=xEst;
%   xe(:,i+1,t)=xEst;
%   Ce{i,t+1}=PEst;
%   Ce{i+1,t}=PEst;
%
```

```
%    end
% end
% Aux=xe(:,:,1)';
% t=linspace(0,steps,steps);
% for i=1:1
%      figure(i);hold on
%      %plot(t,yd(:,i));
%      plot(t,Aux(1:steps,i));hold off
% end
%% figure(13);hold on;
%% %plot3(yd(:,1),yd(:,2),yd(:,3));
%% plot3(Aux(1:steps,1),Aux(1:steps,2),Aux(1:steps,3));hold off;
```

This page is intentionally left blank.

Appendix C

Implementation of calculation of semi empirical calculation for MULLAYA

```
% %this script takes as input a stl file with a 2d profile of the vehicle
% The profile is slice and a function is generated
% Some calculations are require to insure that the center of gravity is at
    coordinates 0,0,0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Wilmer Ariza Ramirez
%Implementarion of calculation fo Preterus Thesis for MULLAYA
%7/11/2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% read Stl file containt file to generate R(x) for integration center of
    figure should be CB
% we export a 3D figure and slide a layer to be used as 2d Template
% file has to be configure as ascii STL

clc; clear vars; close all;
%% REMUS parameters
d=1.60e-1;%vehicle diameter
l=1.47; %vehicle lenght
ro=1030; %Water density
afin=0.131;%corected from preterus
xt=-741.99/1000;%aft end of tail section
xf= -664.99/1000;%aft end of fin section
xf2=-577.092/1000;%%fordward end of fin section
xb2=678.861;%forward end of bow section
Sfin=0.006;%Fin Platform area
Xfin=-620.5/1000;
%% load profile
triangles = read_ascii_stl('MULLAYASimplify.stl',1);
original = triangles;
%triangles = orient_stl(triangles,'z');
triangles = rotate_stl(triangles,'x',90);
slice_height = 0.4;
tic;[movelist, z_slices] = slice_stl_create_path(triangles, slice_height);
    toc;
Path=movelist{2};

plot_slices(movelist,z_slices, 0.0001)

daspect([1 1 1])
%plot capture data
figure(2)
Path(:,1)=(Path(:,1)-678.861)/1000;
Path(:,2)=(Path(:,2)/1000-d/2);
plot(Path(:,1),Path(:,2));
daspect([1 1 1])

%% Interpolation
Xinter = linspace(min(Path(:,1)),max(Path(:,1)),5000);

Path(isnan(Path(:,1)),:) = [];
index=ind2sub([233,1],find(Path(:,2)<0));
Path(index,:) = [];
i=2;
while i<length(Path)
    if round(Path(i-1,1),2)==round(Path(i,1),2)
        Path(i,:)=[];
```

```
    else
        i=i+1;
    end

end

% indices to unique values in column 3
[~, ind] = unique(Path(:,1), 'rows');
% duplicate indices
duplicate_ind = setdiff(1:size(Path(:,1), 1), ind);

[x, index] = unique(Path(:,1));
y=Path(index,2);
I=find(y<0.0005);
x(I)=[];
y(I)=[];
plot(x,y,'*');
Yinter = interp1(x, y, Xinter);


figure(2);
plot(Xinter,Yinter,'*');
daspect([1 1 1])
[m,n]=size(Path);


%% Xudot
betha= 0.23179375;%Blevins aspect ratio
Xudot=-4*betha*ro*pi/3*(d/2)^3;
%% Yvdot
I=find(Xinter>xt & Xinter<xf);

X=Xinter(I);
Y=Yinter(I);
ma_x=pi*ro*Y.^2;
Yvdot=-trapz(X,ma_x);

I=find(Xinter>xf & Xinter<xf2);
X=Xinter(I);
Y=Yinter(I);
maf_x=pi*ro*(afin^2-Y.^2+Y.^4./afin^2);
Yvdot=Yvdot-trapz(X,maf_x);

I=find(Xinter>xf2 & Xinter<xb2);
X=Xinter(I);
Y=Yinter(I);
I=find(isnan(Y) == 1);
X(I)=[];
Y(I)=[];
ma_x=pi*ro*Y.^2;
Yvdot=Yvdot-trapz(X,ma_x);

%% Zwdot
Zwdot=Yvdot;
%% Mwdot
I=find(Xinter>xt & Xinter<xf);

X=Xinter(I);
```

```
Y=Yinter(I);
ma_x=X.*pi*ro.*Y.^2;
Mwdot1=trapz(X,ma_x);

I=find(Xinter>xf & Xinter<xf2);
X=Xinter(I);
Y=Yinter(I);
maf_x=X.*pi*ro.*(afin^2-Y.^2+Y.^4./afin^2);
Mwdot2=trapz(X,maf_x);

I=find(Xinter>xf2 & Xinter<xb2);
X=Xinter(I);
Y=Yinter(I);
I=find(isnan(Y) == 1);
X(I)=[];
Y(I)=[];
ma_x=X.*pi*ro.*Y.^2;
Mwdot3=trapz(X,ma_x);
Mwdot=+Mwdot1-Mwdot2-Mwdot3;
%% Nvdot
Nvdot=-Mwdot;
%% Yrdot
Yrdot=Nvdot;
%% Zqdot
Zqdot=Mwdot;
%% Mqdot
I=find(Xinter>xt & Xinter<xf);

X=Xinter(I);
Y=Yinter(I);
ma_x=X.^2.*pi*ro.*Y.^2;
Mqdot1=trapz(X,ma_x);

I=find(Xinter>xf & Xinter<xf2);
X=Xinter(I);
Y=Yinter(I);
maf_x=X.^2.*pi*ro.*(afin^2-Y.^2+Y.^4./afin^2);
Mqdot2=trapz(X,maf_x);

I=find(Xinter>xf2 & Xinter<xb2);
X=Xinter(I);
Y=Yinter(I);
I=find(isnan(Y) == 1);
X(I)=[];
Y(I)=[];
ma_x=X.^2.*pi*ro.*Y.^2;
Mqdot3=trapz(X,ma_x);
Mqdot=-Mqdot1-Mqdot2-Mqdot3;

%% Nrdot
Nrdot=Mqdot;
%% Kpdot

Kpdot=2/pi*ro*0.1172^4*(xf2-xf);
%% added mass Cross terms

Xwq=Zwdot;
Yura=Xudot;
```

```
Zuqa=-Xudot;
Muwa=-(Zwdot-Xudot);
Nuva=-(Xudot-Yvdot);
Xqq=Zqdot;
Ywp=-Zwdot;
Zvp=Yvdot;
Mvp=-Yrdot;
Nwp=Zqdot;
Xvr=-Yvdot;
Ypq=-Zqdot;
Zrp=Yrdot;
Mrp=(Kpdot-Nrdot);
Npq=-(Kpdot-Mqdot);
Xrr=-Yrdot;
Muqa=-Zqdot;
Nura=Yrdot;

%% Yuvl
Cydbetha=0.003*l/d*180/pi;


Yuvl=-1/2*ro*d^2*Cydbetha;
%% Zuwl
Zuwl=Yuvl;
%% Muwl
Xcp=-0.65*l+678.861/1000;
Muwl=-0.5*ro*d^2*Cydbetha*Xcp;
%% Nuv
Nuvl=Muwl;
%% Fin lift
Span=0.095;
A=0.006;
ARe=2*Span^2/A;
Clalpha=(1/(2*0.9*pi)+1/(pi*ARe))^-1;
xfin=-620/1000;
Yuudr=ro*Clalpha*6.65e-3/2;
Yuvf=-Yuudr;
Zuuds=-ro*Clalpha*6.65e-3/2;
Zuwf=Zuuds;
Yurf=-ro*Clalpha*6.65e-3/2*xfin;
Zuqf=-Yurf;
%% Moment lift
Muuds=ro*Clalpha*6.65e-3/2*-0.638;
Muwf=Muuds;
Nuudr=ro*Clalpha*6.65e-3/2*xfin;
Nuvf=Nuudr;
Muqf=-ro*Clalpha*6.65e-3/2*xfin^2;
Nurf=Muqf;
%% combined terms
Yuv=Yuvl+Yuvf;
Yur=Yura+Yurf;
Zuw=Zuwl+Zuwf;
Zuq=Zuqa+Zuqf;
Muw=Muwa+Muwl+Muwf;
Muq=Muqa+Muqf;
Nuv=Nuva+Nuvl+Nuvf;
Nur=Nura+Nurf;
%% Axial Drag
```

```
Af=pi*(d/2)^2;%Vehicle fron area
Ap=l*d;
Css=6.88e-3;%% from principles of Naval architecture
Cd=Css*pi*Ap/Af*[1+60*(d/l)^3+0.0025*l/d];%aproximation of axial drag
Xuu=-0.5*ro*Cd*Af;

%% Crossflow drag
%% Here i go
I=find(Xinter>xt & Xinter<xb2);

X=Xinter(I);
Y=Yinter(I);
I=find(isnan(Y) == 1);
X(I)=[];
Y(I)=[];
Cdc=0.87;%%https://link.springer.com/content/pdf/10.1007%2Fs00348
    -018-2531-2.pdf
taperratio=38.44/90;
Cdf=0.1+0.7*taperratio;
Zww=-0.5*ro*Cdc*(trapz(X,2*Y))-2.*(0.5*ro*Sfin*Cdf);
Yvv=Zww;
Mww=0.5*ro*Cdc*(trapz(X,2*X.*Y))-2*Xfin*(0.5*ro*Sfin*Cdf);
Nvv=Mww;
Yrr=-0.5*ro*Cdc*(trapz(X,2*X.*abs(X).*Y))-2*Xfin*abs(Xfin)*(0.5*ro*Sfin*Cdf
    );
Zqq=-Yrr;
Mqq=-0.5*ro*Cdc*(trapz(X,2*l.*Y))-2*Xfin*(0.5*ro*Sfin*Cdf);
Nrr=Mqq;
%% Rolling Drag
yvvf=-2.*(0.5*ro*Sfin*Cdf);
Kpp=yvvf*afin^3;


%% Inertia
In=[0.083946018 0 0;
    0 3.084445738 0;
    0 0 3.062612431];
m=24.4;%kg
W=239.364;%newtons
B=246.2;% Newtons
```

Appendix D

MULLAYA calculated Matlab model

```
function xdot= MULLAYA(t,x,n,delta_s,delta_r)
% Simulation algorithm for MULLAYA this code is based in:
% Prestero, T., 2001. Verification of a six-degree of freedom simulation
% model for the REMUS autonomous underwater vehicle (Doctoral dissertation,
% Massachusetts Institute of Technology and Woods Hole Oceanographic
% Institution).
% This code has been modified to run with ode45 solver
% Programmed by: Wilmer Ariza Ramirez
%University of Tasmania
%Australian Maritime College

%                 -----F1-----------------------------------
%           |   /             ||                            ||    \
%           |==       F3      ||        Remus               ||     |
%           |   \-------------||------------------------    ||---/
%                       F2
% V=[u,v,w,p,q,r,x,y,z,phi,psi,theta];


% TERMS
% -------------------------------------------------
%STATE VECTOR:
%
% x = (u v w p q r xpos ypos zpos phi theta psi) ,
% Body-referenced Coordinates
% u  = Surge velocity  [m/s]
% v  = Sway velocity   [m/s]
% w  = Heave velocity  [m/s]
% p  = Roll rate       [rad/s]
% q  = Pitch rate      [rad/s]
% r  = Yaw rate        [rad/s]
%
% Earth-fixed coordinates
% xpos  = Position in x-direction [m]
% ypos  = Position in y-direction [m]
% zpos  = Position in z-direction [m]
% phi   = Roll angle              [rad]
% theta = Pitch angle             [rad]
% psi   = Yaw angle               [rad]
%
%INPUT VECTOR
% ui = [n delta_s delta_r]'
%  Control Fin Angles
%   n= shaft speed RPM
%   delta_s = angle of stern planes  [rad]
%   delta_r = angle of rudder planes [rad]

% Get state variables
u     = x(1);
v     = x(2);
w     = x(3);
p     = x(4);
q     = x(5);
r     = x(6);
phi   = x(10);
theta = x(11);
psi   = x(12);
ui = [n(t) delta_s(t) delta_r(t)];
```

```
if ui(1)<0
    ui(1)=0;
end
% tao_u=ui(1);
% tao_q=ui(2);
% tao_r=ui(3);

% Get control inputs
delta_s = -ui(2)*pi/180;
delta_r = ui(3)*pi/180;

% 45 degree maximum rudder angle
delta_max = 45 * pi/180;

% Check control inputs (useful later)
% if abs(delta_s) > delta_max && delta_s < delta_max
%     delta_s = sign(delta_s) * delta_max;
% end
%
% if abs(delta_r) > delta_max
%     delta_r = sign(delta_r) * delta_max;
% end

% Initialize elements of coordinate system transform matrix
% ------------------------------------------
    ------------------------------------------
c1 = cos(phi);
c2 = cos(theta);
c3 = cos(psi);
s1 = sin(phi);
s2 = sin(theta);
s3 = sin(psi);
t2 = tan(theta);

% ------------------------------------------
% Vehicle Parameters and Coefficients
% ------------------------------------------
% ------------------------------------------
% Vehicle Parameters and Coefficients
% ------------------------------------------
W=239.364;%newtons
B=246.2;% Newtons

g = 9.81; % Force of gravity
m = W/g; % Mass of vehicle

Xuu   = -2.8808;    % Axial Drag
Xwq   = -29.6403;  % Added mass cross-term
Xqq   = 0.6579;    % Added mass cross-term
Xvr   =  29.6403;  % Added mass cross-term
Xrr   = 0.6579;    % Added mass cross-term
Yvv   = -95.3687;  % Cross-flow drag
Yrr   =  -2.4529; % Cross-flow drag
Yuv   = -32.9370;  % Body lift force and fin lift
Ywp   =  29.6403;  % Added mass cross-term
Yur   =  7;    % Added mass cross-term and fin lift
Ypq   =  -0.6579;    % Added mass cross-term
Zww   = -95.3687;  % Cross-flow drag
```

```
Zqq    = 2.4529; % Cross-flow drag
Zuw    = -32.9370;  % Body lift force and fin lift
Zuq    = -7;    % Added mass cross-term and fin lift
Zvp    = -29.6403;  % Added mass cross-term
Zrp    =  -0.6579;    % Added mass cross-term

% Center of Gravity wrt Origin at CB
xg = 0;
yg = 0;
zg = 1.96e-2;

% Control Fin Coefficients
Yuudr =  12.1167;
Nuudr = -7.5123;
Zuuds = -12.1167; % Fin Lift Force

% Center of Buoyancy wrt Origin at Vehicle Nose
xb = 0;%-6.11e-1;
yb =  0;
zb =  0;
n=ui(1)/60*2*pi;
% Propeller Terms
Xprop =  3e-4*n*abs(n);
Kpp    = -1.3e-1;  % Rolling resistance
Kprop = -2.242e-05*n*abs(n);%-5.43e-1; % Propeller Torque
Kpdot = -7.04e-2; % Added mass

% Cross flow drag and added mass terms
Mww    =  6.9559; % Cross-flow drag
Mqq    = -135.0375; % Cross-flow drag
Mrp    =  5.0635; % Added mass cross-term
Muq    = -5.3156; % Added mass cross term and fin lift
Muw    =   27.1576; % Body and fin lift and munk moment
Mwdot = 0.6579; % Added mass
Mvp    = 0.6579; % Added mass cross term
Muuds = -7.7304; % Fin lift moment
Nvv    = 6.9559; % Cross-flow drag
Nrr    = -135.03; % Cross-flow drag
Nuv    = -30.8809; % Body and fin lift and munk moment
Npq    = -5.0635; % Added mass cross-term

% Moments of Inertia wrt Origin at CB
Ixx   = 0.083946018;
Iyy   = 3.084445738;
Izz   = 3.062612431;

Nwp = 0.6579; % Added mass cross-term
Nur = -5.3153; % Added mass cross term and fin lift

% Non-linear Moments Coefficients
Xudot = -0.5120; % Added mass
Yvdot = -29.6403;  % Added mass
Nvdot =  -0.6579;    % Added mass
Mwdot = 0.6579;    % Added mass
Mqdot = -5.0526;    % Added mass
Zqdot = 0.6579;    % Added mass
Zwdot = -29.6403;  % Added mass
Yrdot = - 0.6579;    % Added mass
```

```
Nrdot = -5.0526;     % Added mass
% Set total forces from equations of motion
% ------------------------------------------------------------
    ----------------------------

X = -(W-B)*sin(theta) + Xuu*u*abs(u) + (Xwq-m)*w*q + (Xqq + m*xg)*q^2  ...
    + (Xvr+m)*v*r + (Xrr + m*xg)*r^2 -m*yg*p*q - m*zg*p*r  ...
    + ui(1) ;

Y = (W-B)*cos(theta)*sin(phi) + Yvv*v*abs(v) + Yrr*r*abs(r) + Yuv*u*v  ...
    + (Ywp+m)*w*p + (Yur-m)*u*r - (m*zg)*q*r + (Ypq - m*xg)*p*q  ...
    ;

Z = (W-B)*cos(theta)*cos(phi) + Zww*w*abs(w) + Zqq*q*abs(q)+ Zuw*u*w  ...
    + (Zuq+m)*u*q + (Zvp-m)*v*p + (m*zg)*p^2 + (m*zg)*q^2  ...
    + (Zrp - m*xg)*r*p   ;

K = -(yg*W-yb*B)*cos(theta)*cos(phi) - (zg*W-zb*B)*cos(theta)*sin(phi)  ...
    + Kpp*p*abs(p) - (Izz- Iyy)*q*r - (m*zg)*w*p + (m*zg)*u*r  + Kprop;

M = -(zg*W-zb*B)*sin(theta) - (xg*W-xb*B)*cos(theta)*cos(phi) + Mww*w*abs(w
    )  ...
    + Mqq*q*abs(q) + (Mrp - (Ixx-Izz))*r*p + (m*zg)*v*r - (m*zg)*w*q  ...
    + (Muq - m*xg)*u*q + Muw*u*w + (Mvp + m*xg)*v*p  ...
    + ui(2);

N = -(xg*W-xb*B)*cos(theta)*sin(phi) - (yg*W-yb*B)*sin(theta)  ...
    + Nvv*v*abs(v) + Nrr*r*abs(r) + Nuv*u*v  ...
    + (Npq - (Iyy- Ixx))*p*q + (Nwp - m*xg)*w*p + (Nur + m*xg)*u*r  ...
    +  ui(3) ;


FORCES = [X Y Z K M N]';

% Accelerations Matrix (Prestero Thesis page 46)
Amat = [(m - Xudot) 0                 0                 0               m*zg
      -m*yg;
        0             (m - Yvdot)     0                 -m*zg           0
      (m*xg - Yrdot);
        0             0               (m - Zwdot)       m*yg            (-m*xg -
    Zqdot) 0;
        0             -m*zg           m*yg              (Ixx - Kpdot)   0
      0;
        m*zg          0               (-m*xg - Mwdot)   0               (Iyy -
    Mqdot)    0;
        -m*yg         (m*xg - Nvdot)  0                 0               0
      (Izz - Nrdot)];

% Inverse Mass Matrix
Minv = inv(Amat);

% Derivatives
xdot = ...
    [Minv(1,1)*X + Minv(1,2)*Y + Minv(1,3)*Z + Minv(1,4)*K + Minv(1,5)*M +
    Minv(1,6)*N
     Minv(2,1)*X + Minv(2,2)*Y + Minv(2,3)*Z + Minv(2,4)*K + Minv(2,5)*M +
    Minv(2,6)*N
     Minv(3,1)*X + Minv(3,2)*Y + Minv(3,3)*Z + Minv(3,4)*K + Minv(3,5)*M +
    Minv(3,6)*N
```

```
  Minv(4,1)*X + Minv(4,2)*Y + Minv(4,3)*Z + Minv(4,4)*K + Minv(4,5)*M +
Minv(4,6)*N
  Minv(5,1)*X + Minv(5,2)*Y + Minv(5,3)*Z + Minv(5,4)*K + Minv(5,5)*M +
Minv(5,6)*N
  Minv(6,1)*X + Minv(6,2)*Y + Minv(6,3)*Z + Minv(6,4)*K + Minv(6,5)*M +
Minv(6,6)*N
  c3*c2*u + (c3*s2*s1-s3*c1)*v + (s3*s1+c3*c1*s2)*w
  s3*c2*u + (c1*c3+s1*s2*s3)*v + (c1*s2*s3-c3*s1)*w
    -s2*u +            c2*s1*v +            c1*c2*w
        p +            s1*t2*q +            c1*t2*r
                          c1*q -               s1*r
                       s1/c2*q +            c1/c2*r] ;
```

BIBLIOGRAPHY

[1] D Richard Blidberg. The development of autonomous underwater vehicles (auv); a brief summary. In *Ieee Icra*, volume 4, page 1, 2001.

[2] M. Breivik and T. I. Fossen. Guidance-based path following for autonomous underwater vehicles. *Oceans 2005, Vols 1-3*, pages 2807–2814, 2005. ISSN 0197-7385. URL <GotoISI>://WOS:000238978702159.

[3] F Aguirre, S Vargas2 D Valdés, and J Tornero. State of the art of parameters for mechanical design of an autonomous underwater vehicle. *International Journal of Oceans and Oceanography*, 11(1):89–103, 2017.

[4] John J. Leonard, Andrew A. Bennett, Christopher M. Smith, Hans Jacob, and S. Feder. Autonomous underwater vehicle navigation. In *MIT Marine Robotics Laboratory Technical Memorandum*, 1998.

[5] TN Ranjan, Arun Nherakkol, and Gajanan Navelkar. Navigation of autonomous underwater vehicle using extended kalman filter. In *FIRA RoboWorld Congress*, pages 1–9. Springer, 2010.

[6] A Okamoto, JJ Feeley, DB Edwards, and RW Wall. Robust control of a platoon of underwater autonomous vehicles. In *Oceans' 04 MTS/IEEE Techno-Ocean'04 (IEEE Cat. No. 04CH37600)*, volume 1, pages 505–510. IEEE, 2004.

[7] W. Naeem, R. Sutton, and S.M. Ahmad. Lqg/ltr control of an autonomous underwater vehicle using a hybrid guidance law. *IFAC Proceedings Volumes*, 36(4):31 – 36, 2003. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)36653-3. URL http://www.sciencedirect.com/science/article/pii/S1474667017366533. IFAC Workshop on Guidance and Control of Underwater Vehicles 2003, Newport, South Wales, UK, 9-11 April 2003.

[8] Xiao Liang, Lei Wan, James I.R. Blake, R. Ajit Shenoi, and Nicholas Townsend. Path following of an underactuated auv based on fuzzy backstepping sliding mode control. *International Journal of Advanced Robotic Systems*, 13(3):122, 2016. doi: 10.5772/64065. URL https://doi.org/10.5772/64065.

[9] J. Wang, C. Wang, Y. Wei, and C. Zhang. Three-dimensional path following of an underactuated auv based on neuro-adaptive command filtered backstepping control. *IEEE Access*, pages 1–1, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2883081.

[10] James C Kinsey, Ryan M Eustice, and Louis L Whitcomb. A survey of underwater vehicle navigation: Recent advances and new challenges. In *IFAC Conference of Manoeuvering and Control of Marine Craft*, volume 88, pages 1–12, 2006.

[11] Jørgen Lorentz and J Yuh. A survey and experimental study of neural network auv control. In *Proceedings of Symposium on Autonomous Underwater Vehicle Technology*, pages 109–116. IEEE, 1996.

[12] Juš Kocijan. *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer, 2016. ISBN 3319210203.

[13] Wilmer Ariza Ramirez, Zhi Quan Leong, Hung Nguyen, and Shantha Gamini Jayasinghe. Non-parametric dynamic system identification of ships using multi-output gaussian processes. *Ocean Engineering*, 166:26–36, 2018. ISSN 0029-8018. doi: https://doi.org/10.

1016/j.oceaneng.2018.07.056. URL http://www.sciencedirect.com/science/article/pii/S0029801818314094.

[14] Frode Olsgard and John S Gray. A comprehensive analysis of the effects of offshore oil and gas exploration and production on the benthic communities of the norwegian continental shelf. *Marine Ecology Progress Series*, 122:277–306, 1995.

[15] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[16] Kristin Y Pettersen and Henk Nijmeijer. Underactuated ship tracking control: theory and experiments. *International Journal of Control*, 74(14):1435–1446, 2001.

[17] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[18] Martin A Abkowitz. Lectures on ship hydrodynamics–steering and manoeuvrability. Technical report, 1964.

[19] Kyoungho Son and Kensaku Nomoto. On the coupled motion of steering and rolling of a high speed container ship. *Journal of the Society of Naval Architects of Japan*, 1981(150): 232–244, 1981.

[20] CG Källström, Karl Johan Åström, NE Thorell, J Eriksson, and L Sten. Adaptive autopilots for large tankers. *IFAC Proceedings Volumes*, 11(1):477–484, 1978.

[21] Thor I Fossen et al. *Guidance and control of ocean vehicles*, volume 199. Wiley New York, 1994.

[22] Yaseen Adnan Ahmed and Kazuhiko Hasegawa. Automatic ship berthing using artificial neural network trained by consistent teaching data using nonlinear programming method. *Engineering Applications of Artificial Intelligence*, 26(10):2287–2304, 2013.

[23] Richard Evelyn Donohue Bishop and Amie Gladys Parkinson. On the planar motion mechanism used in ship model testing. *Phil. Trans. R. Soc. Lond. A*, 266(1171):35–61, 1970.

[24] F Stern, K Agdrup, SY Kim, AC Hochbaum, KP Rhee, FHHA Quadvlieg, P Perdon, Takanori Hino, R Broglia, and J Gorski. Experience from simman 2008—the first workshop on verification and validation of ship maneuvering simulation methods. *Journal of Ship Research*, 55(2):135–147, 2011.

[25] Karl Johan Åström and Claes G Källström. Identification of ship steering dynamics. *Automatica*, 12(1):9–22, 1976.

[26] HL Brinati and A Rios Neto. Application of the extended kalman filtering to the identification of ship hydrodynamic coefficients. In *Proceedings of the Third Brazilian Congress of Mechanical Engineering*, volume 100, pages 791–804, 1975.

[27] WW Zhou and M Blanke. Nonlinear recursive prediction error method applied to identification of ship steering dynamics. 1987.

[28] Hyeon Kyu Yoon and Key Pyo Rhee. Identification of hydrodynamic coefficients in ship maneuvering equations of motion by estimation-before-modeling technique. *Ocean Engineering*, 30(18):2379–2404, 2003.

[29] Manuel Haro Casado, Ramón Ferreiro, and FJ Velasco. Identification of nonlinear ship model parameters based on the turning circle test. *Journal of ship research*, 51(2):174–181, 2007.

[30] Mahmoud R Haddara and Yie Wang. Parametric identification of manoeuvring models for ships. *International Shipbuilding Progress*, 46(445):5–27, 1999.

[31] WL Luo and ZJ Zou. Parametric identification of ship maneuvering models by using support vector machines. *Journal of Ship Research*, 53(1):19–30, 2009.

[32] Ozan Irsoy and Claire Cardie. Deep recursive neural networks for compositionality in language. In *Advances in neural information processing systems*, pages 2096–2104, 2014.

[33] L Moreira and C Guedes Soares. Recursive neural network model of catamaran manoeuvring. *International Journal of Maritime Engineering*, 154:A121–A130, 2012.

[34] Xue-gang Wang, Zao-jian Zou, Long Yu, and Wei Cai. System identification modeling of ship manoeuvring motion in 4 degrees of freedom based on support vector machines. *China Ocean Engineering*, 29(4):519–534, 2015.

[35] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[36] Daniel G Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6): 119–139, 1951.

[37] Juš Kocijan, Agathe Girard, Blaž Banko, and Roderick Murray-Smith. Dynamic systems identification with gaussian processes. *Mathematical and Computer Modelling of Dynamical Systems*, 11(4):411–424, 2005.

[38] K Ažman and Jus Kocijan. Dynamical systems identification using gaussian process models with incorporated local models. *Engineering Applications of Artificial Intelligence*, 24(2): 398–408, 2011.

[39] Mauricio Alvarez and Neil D Lawrence. Sparse convolved gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pages 57–64, 2009.

[40] Dave Higdon. Space and space-time modeling using process convolutions. In *Quantitative methods for current environmental issues*, pages 37–56. Springer, 2002.

[41] Phillip Boyle and Marcus Frean. Dependent gaussian processes. In *Advances in neural information processing systems*, pages 217–224, 2005.

[42] Mauricio Alvarez and Neil Lawrence. Multiple output gaussian processes in matlab. https://github.com/SheffieldML/multigp, 2014.

[43] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[44] Benedetto Allotta, Riccardo Costanzi, Luca Pugi, Alessandro Ridolfi, and Andrea Rindi. *Fast Calibration Procedure of the Dynamic Model of an Autonomous Underwater Vehicle from a Reduced Set of Experimental Data*, pages 317–326. Springer, 2017.

[45] Richard Evelyn Donohue Bishop and Amie Gladys Parkinson. On the planar motion mechanism used in ship model testing. *Phil. Trans. R. Soc. Lond. A*, 266(1171):35–61, 1970. ISSN 0080-4614.

[46] Francisco J Velasco, Elías Revestido Herrero, Francisco J Lastra Santos, José María Riola Rodríguez, Juan Jesús Díaz Hernández, and Luis M Vega Antolín. Measurements of hydrodynamic parameters and control of an underwater torpedo-shaped vehicle. *IFAC-PapersOnLine*, 48(2):167–172, 2015. ISSN 2405-8963.

[47] Hiroyoshi Suzuki, Junki Sakaguchi, Tomoya Inoue, Yoshitaka Watanabe, and Hiroshi Yoshida. Evaluation of methods to estimate hydrodynamic force coefficients of underwater vehicle based on cfd. *IFAC Proceedings Volumes*, 46(33):197–202, 2013. ISSN 1474-6670.

[48] Amit Tyagi and Debabrata Sen. Calculation of transverse hydrodynamic coefficients using computational fluid dynamic approach. *Ocean Engineering*, 33(5-6):798–809, 2006. ISSN 0029-8018.

[49] ZQ Leong, D Ranmuthugala, AL Forrest, and J Duffy. Numerical investigation of the hydrodynamic interaction between two underwater bodies in relative motion. *Applied Ocean Research*, 51:14–24, 2015. ISSN 0141-1187.

[50] S Khalil Shariati and S Hossein Mousavizadegan. The effect of appendages on the hydrodynamic characteristics of an underwater vehicle near the free surface. *Applied Ocean Research*, 67:31–43, 2017. ISSN 0141-1187.

[51] KP Rhee, SY Lee, and YJ Sung. Estimation of manoeuvring coefficients from pmm test by genetic algorithm. In *Procedings of International Symposium and Workshop on Force Acting on a Manoeuvring Vessel, Val de Reuil, France*, pages 77–87, 1998.

[52] Andrew Ross, Thor I Fossen, and Tor Arne Johansen. Identification of underwater vehicle hydrodynamic coefficients using free decay tests. *IFAC Proceedings Volumes*, 37(10):363–368, 2004. ISSN 1474-6670.

[53] Ehsan Shahinfar, Mohammad Bozorg, and Mohsen Bidoky. Parameter estimation of an auv using the maximum likelihood method and a kalman filter with fading memory. *IFAC Proceedings Volumes*, 43(16):1–6, 2010. ISSN 1474-6670.

[54] Mohammad Taghi Sabet, Pouria Sarhadi, and Mostafa Zarini. Extended and unscented kalman filters for parameter estimation of an autonomous underwater vehicle. *Ocean Engineering*, 91:329–339, 2014. ISSN 0029-8018.

[55] T. Perez and T. I. Fossen. Practical aspects of frequency-domain identification of dynamic models of marine structures from hydrodynamic data. *Ocean Engineering*, 38(2-3):426–435, 2011. ISSN 0029-8018. doi: 10.1016/j.oceaneng.2010.11.004. URL <GotoISI>://WOS:000287629600014.

[56] Pepijn WJ Van De Ven, Tor A Johansen, Asgeir J Sørensen, Colin Flanagan, and Daniel Toal. Neural network augmented identification of underwater vehicle models. *Control Engineering Practice*, 15(6):715–725, 2007. ISSN 0967-0661.

[57] XU Feng, Zao-jian ZOU, Jian-chuan YIN, and CAO Jian. Parametric identification and sensitivity analysis for autonomous underwater vehicles in diving plane. *Journal of Hydrodynamics, Ser. B*, 24(5):744–751, 2012. ISSN 1001-6058.

[58] Pepijn van de Ven, Colin Flanagan, and Daniel Toal. Identification of underwater vehicle dynamics with neural networks. In *OCEANS'04. MTTS/IEEE TECHNO-OCEAN'04*, volume 3, pages 1198–1204. IEEE, 2004. ISBN 0780386698.

[59] Feng Xu, Zao-Jian Zou, Jian-Chuan Yin, and Jian Cao. Identification modeling of underwater vehicles' nonlinear dynamics based on support vector machines. *Ocean Engineering*, 67:68–76, 2013. ISSN 0029-8018.

[60] VS Kodogiannis, Paulo JG Lisboa, and J Lucas. Neural network modelling and control for underwater vehicles. *Artificial intelligence in Engineering*, 10(3):203–212, 1996. ISSN 0954-1810.

[61] Bilal Wehbe, Marc Hildebrandt, and Frank Kirchner. Experimental evaluation of various machine learning regression methods for model identification of autonomous underwater vehicles. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4885–4890. IEEE, 2017. ISBN 150904633X.

[62] Juš Kocijan and Alexandra Grancharova. Gaussian process modelling case study with multiple outputs. *Comptes rendus de l'Académie bulgare des Sciences*, 63:601–607, 2010.

[63] Mauricio A Álvarez and Neil D Lawrence. Computationally efficient convolved multiple output gaussian processes. *Journal of Machine Learning Research*, 12(May):1459–1500, 2011.

[64] Jing Zhao and Shiliang Sun. Variational dependent multi-output gaussian process dynamical systems. *The Journal of Machine Learning Research*, 17(1):4134–4169, 2016. ISSN 1532-4435.

[65] Mauricio Alvarez and Neil D Lawrence. Sparse convolved gaussian processes for multi-output regression. In *Advances in neural information processing systems*, pages 57–64, 2009.

[66] Timothy Prestero. *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle*. Thesis, 2001.

[67] Morton Gertler and Grant R Hagen. Standard equations of motion for submarine simulation. Technical report, DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BETHESDA MD, 1967.

[68] Jinhyun Kim and Wan Kyun Chung. Accurate and practical thruster modeling for underwater vehicles. *Ocean Engineering*, 33(5-6):566–586, 2006.

[69] Mark Ebden. Gaussian processes for regression: A quick introduction. *The Website of Robotics Research Group in Department on Engineering Science, University of Oxford*, 2008.

[70] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

[71] Christopher K Williams and Carl Edward Rasmussen. Gaussian processes for machine learning. *the MIT Press*, 2(3):4, 2006.

[72] Raewyn Hall and Stuart Anstee. Trim calculation methods for a dynamical model of the remus 100 autonomous underwater vehicle. Report, DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA) MARITIME OPERATIONS DIV, 2011.

[73] B Allotta, A Caiti, L Chisci, R Costanzi, F Di Corato, C Fantacci, D Fenucci, E Meli, and A Ridolfi. An unscented kalman filter based navigation algorithm for autonomous underwater vehicles. *Mechatronics*, 39:185–195, 2016. ISSN 0957-4158.

[74] Derek A Fong and Nicole L Jones. Evaluation of auv-based adcp measurements. *Limnology and Oceanography: methods*, 4(3):58–67, 2006. ISSN 1541-5856.

[75] William J Van Trump and Matthew J McHenry. The morphology and mechanical sensitivity of lateral line receptors in zebrafish larvae (danio rerio). *Journal of Experimental Biology*, 211(13):2105–2115, 2008. ISSN 0022-0949.

[76] Juan F Fuentes-Pérez, Kaia Kalev, Jeffrey A Tuhtan, and Maarja Kruusmaa. Underwater vehicle speedometry using differential pressure sensors: Preliminary results. In *Autonomous Underwater Vehicles (AUV), 2016 IEEE/OES*, pages 156–160. IEEE, 2016. ISBN 1509024425.

[77] W. Wang, Li Yuan, X. Zhang, Wang Chen, S. Chen, and G. Xie. Speed evaluation of a freely swimming robotic fish with an artificial lateral line. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4737–4742, 2016. doi: 10.1109/ICRA.2016.7487675.

[78] Taavi Salumäe and Maarja Kruusmaa. Flow-relative control of an underwater robot. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 469(2153), 2013. doi: 10.1098/rspa.2012.0671. URL http://rspa.royalsocietypublishing.org/content/royprsa/469/2153/20120671.full.pdf.

[79] A. G. P. Kottapalli, M. Asadnia, J. M. Miao, C. W. Tan, G. Barbastathis, and M. Triantafyllou. Polymer mems pressure sensor arrays for fish-like underwater sensing applications. *IET Micro & Nano Letters*, 7(12):1189–1192, 2012. ISSN 1750-0443. doi: 10.1049/mnl.2012.0604.

[80] Vicente Ignacio Fernandez, Stephen M Hou, Franz S Hover, Jeffrey H Lang, and Michael S Triantafyllou. Lateral-line inspired mems-array pressure sensing for passive underwater navigation. Report, Massachusetts Institute of Technology. Sea Grant College Program, 2007.

[81] Saunvit Pandya, Yingchen Yang, Douglas L Jones, Jonathan Engel, and Chang Liu. Multisensor processing algorithms for underwater dipole localization and tracking using mems artificial lateral-line sensors. *EURASIP Journal on Applied Signal Processing*, 2006:199–199, 2006. ISSN 1110-8657.

[82] *MPXV5010 Datasheet*. NXP, 2012. Rev. 10.0.

[83] NXP. *AN1646*, 2005.

[84] Sheng Chen, SA Billings, and PM Grant. Non-linear system identification using neural networks. *International journal of control*, 51(6):1191–1214, 1990. ISSN 0020-7179.

[85] Hava T Siegelmann, Bill G Horne, and C Lee Giles. Computational capabilities of recurrent narx neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):208–215, 1997. ISSN 1083-4419.

[86] Carl Edward Rasmussen. *Evaluation of Gaussian processes and other methods for non-linear regression*. Citeseer, 1999. ISBN 0612283003.

[87] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012. ISBN 1461207452.

[88] Carl Edward Rasmussen. *Gaussian processes in machine learning*, pages 63–71. Springer, 2004.

[89] Wilmer Ariza Ramirez, Zhi Quan Leong, Hung Nguyen, and Shantha Gamini Jayasinghe. Position estimation for underwater vehicles using unscented kalman filter with gaussian process prediction. *Underwater Technology*, 36(2), 2019. ISSN 1756-0543. doi: 10.3723/ut.36.015.

[90] GD Dunlap and Henry H Shufeldt. Dutton's navigation and piloting. *Dutton's navigation and piloting., by Dunlap, GD; Shufeldt, HH. Annapolis, MD (USA): United States Naval Institute, 715 p.*, 1969.

[91] Morris M Kuritsky and Murray S Goldstein. *Inertial navigation*, pages 96–116. Springer, 1990.

[92] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. ISSN 0021-9223.

[93] B. Armstrong, E. Wolbrecht, and D. B. Edwards. Auv navigation in the presence of a magnetic disturbance with an extended kalman filter. In *OCEANS 2010 IEEE - Sydney*, pages 1–6, 2010. doi: 10.1109/OCEANSSYD.2010.5603905.

[94] B. Allotta, A. Caiti, R. Costanzi, F. Fanelli, D. Fenucci, E. Meli, and A. Ridolfi. A new auv navigation system exploiting unscented kalman filter. *Ocean Engineering*, 113:121–132, 2016. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2015.12.058. URL http://www.sciencedirect.com/science/article/pii/S0029801815007271.

[95] R. P. Vio, R. Cristi, and K. B. Smith. Near real-time improved uuv positioning through channel estimation - the unscented kalman filter approach. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–7. doi: 10.1109/OCEANS.2016.7761087.

[96] B. Allotta, A. Caiti, L. Chisci, R. Costanzi, F. Di Corato, C. Fantacci, D. Fenucci, E. Meli, and A. Ridolfi. An unscented kalman filter based navigation algorithm for autonomous underwater vehicles. *Mechatronics*, 39:185–195, 2016. ISSN 0957-4158. doi: https://doi.org/10.1016/j.mechatronics.2016.05.007. URL http://www.sciencedirect.com/science/article/pii/S095741581630037X.

[97] R. Turner and C. E. Rasmussen. Model based learning of sigma points in unscented kalman filtering. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 178–183. ISBN 1551-2541. doi: 10.1109/MLSP.2010.5589003.

[98] Jan Mandel. A brief tutorial on the ensemble kalman filter. *arXiv preprint arXiv:0901.3725*, 2009.

[99] D. Loebis, R. Sutton, and J. Chudley. A fuzzy kalman filter for accurate navigation of an autonomous underwater vehicle. *IFAC Proceedings Volumes*, 36(4):157–162, 2003. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)36674-0. URL http://www.sciencedirect.com/science/article/pii/S1474667017366740.

[100] Ngatini, Erna Apriliani, and Hendro Nurhadi. Ensemble and fuzzy kalman filter for position estimation of an autonomous underwater vehicle based on dynamical system of auv motion. *Expert Systems with Applications*, 68:29–35, 2017. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.10.003. URL http://www.sciencedirect.com/science/article/pii/S0957417416305346.

[101] Rodrigo Telles da Silva Vale, Ettore Apolonio de Barros, and Thiago de Castro Martins. Gpu-accelerated monte carlo localization for underwater robots. *IFAC-PapersOnLine*, 48 (2):76–81, 2015. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2015.06.013. URL http://www.sciencedirect.com/science/article/pii/S2405896315002529.

[102] He Zhang, Yu-ru Xu, and Hao-peng Cai. Using cfd software to calculate hydrodynamic coefficients. *Journal of Marine Science and Application*, 9(2):149–155, 2010. ISSN 1671-9433.

[103] Debabrata Sen. A study on sensitivity of manoeuvrability performance on the hydrodynamic coefficients for submerged bodies. *J Ship Res*, 45(3):186–196, 2000.

[104] J. Ko, D. J. Kleint, D. Fox, and D. Haehnelt. Gp-ukf: Unscented kalman filters with gaussian process prediction and observation models. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1901–1907, 2007. ISBN 2153-0858. doi: 10.1109/IROS.2007.4399284.

[105] Morton Gertler and Grant R Hagen. Standard equations of motion for submarine simulation. Report, DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BETHESDA MD, 1967.

[106] Jeffrey K Uhlmann. *Dynamic map building and localization: New theoretical foundations*. Thesis, 1995.

[107] Mark Ebden et al. Gaussian processes for regression: A quick introduction. *The Website of Robotics Research Group in Department on Engineering Science, University of Oxford*, 91:424–436, 2008.

[108] Xianbo Xiang, Caoyang Yu, and Qin Zhang. Robust fuzzy 3d path following for autonomous underwater vehicle subject to uncertainties. *Computers and Operations Research*, 84:165 – 177, 2017. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2016.09.017. URL http://www.sciencedirect.com/science/article/pii/S0305054816302374.

[109] Marc Peter Deisenroth, Marco F Huber, and Uwe D Hanebeck. Analytic moment-based gaussian process filtering. In *Proceedings of the 26th annual international conference on machine learning*, pages 225–232. ACM, 2009.

[110] Josefine Severholt. Generic 6-dof added mass formulation for arbitrary underwater vehicles based on existing semi-empirical methods, 2017.

[111] Jesse Stuart Geisbert. *Hydrodynamic modeling for autonomous underwater vehicles using computational and semi-empirical methods*. PhD thesis, Virginia Tech, 2007.

[112] Anthony Travers. Simulation trial results for the cooperative autonomous underwater vehicle demonstration (sa15). Technical report, DEFENSE SCIENCE AND TECHNOLOGY ORGANIZATION VICTORIA (AUSTRALIA) MARITIME PLATFORMS DIV, 2013.

[113] C Madden. An evaluation of potential operating systems for autonomous underwater vehicles. Technical report, DEFENSE SCIENCE AND TECHNOLOGY ORGANIZATION VICTORIA (AUSTRALIA) MARITIME PLATFORMS DIV, 2013.

[114] Sunil Bhandari, 2018. URL https://au.mathworks.com/matlabcentral/fileexchange/62113-slice_stl_create_path-triangles-slice_height.

[115] John Nicholas Newman and L Landweber. Marine hydrodynamics, 1978.

[116] Robert D Blevins and R Plunkett. *Formulas for natural frequency and mode shape*. American Society of Mechanical Engineers, 1980. ISBN 0021-8936.

[117] Sighard F Hoerner and Henry V Borst. Fluid-dynamic lift, practical information on aerodynamic and hydrodynamic lift. Technical report, BORST (HENRY V) AND ASSOCIATES WAYNE PA, 1975.

[118] Michael S Triantafyllou. Maneuvering and control of surface and underwater vehicles. 2004.

[119] Zaopeng Dong, Lei Wan, Yueming Li, Tao Liu, Jiayuan Zhuang, and Guocheng Zhang. Point stabilization for an underactuated auv in the presence of ocean currents. *International Journal of Advanced Robotic Systems*, 12(7):100, 2015. doi: 10.5772/61037. URL https://doi.org/10.5772/61037.

[120] F. Alonge, F. D'Ippolito, and F. M. Raimondi. Trajectory tracking of underactuated underwater vehicles. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, volume 5, pages 4421–4426 vol.5, Dec 2001. doi: 10.1109/CDC.2001.980898.

[121] X. Xiang, L. Lapierre, C. Liu, and B. Jouvencel. Path tracking: Combined path following and trajectory tracking for autonomous underwater vehicles. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3558–3563, Sept 2011. doi: 10.1109/IROS.2011.6094949.

[122] Ahmad Forouzantabar, Babak Gholami, and Mohammad Azadi. Adaptive neural network control of autonomous underwater vehicles. *World Academy of Science, Engineering and Technology*, 6(7):304–309, 2012.

[123] K. D. Do, J. Pan, and Z. P. Jiang. Robust and adaptive path following for underactuated autonomous underwater vehicles. *Ocean Engineering*, 31(16):1967–1997, 2004. ISSN 0029-8018. doi: 10.1016/j.oceaneng.2004.04.006. URL <GotoISI>://WOS:000224890800003.

[124] Xianbo Xiang, Lionel Lapierre, and Bruno Jouvencel. Smooth transition of auv motion control: From fully-actuated to under-actuated configuration. *Robotics and Autonomous Systems*, 67:14 – 22, 2015. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2014.09.024. URL http://www.sciencedirect.com/science/article/pii/S0921889014002097. Advances in Autonomous Underwater Robotics.

[125] C. Roper. Using the gavia auv system to locate and document munitions dumped at sea., 2007. URL http://underwatermunitions.org/wp-content/uploads/2016/03/Gaviapresentation_21Aug07.pdf.

[126] Sigurd Andreas Holsen. Dune: Unified navigation environment for the remus 100 auv-implementation, simulator development, and field experiments. Master's thesis, NTNU, 2015.

[127] R. Rout and B. Subudhi. Narmax self-tuning controller for line-of-sight-based waypoint tracking for an autonomous underwater vehicle. *IEEE Transactions on Control Systems Technology*, 25(4):1529–1536, July 2017. ISSN 1063-6536. doi: 10.1109/TCST.2016.2613969.

[128] Mansour Ataei and Aghil Yousefi-Koma. Three-dimensional optimal path planning for waypoint guidance of an autonomous underwater vehicle. *Robotics and Autonomous Systems*, 67:23–32, 2015.

[129] S Saravanakumar and T Asokan. Waypoint guidance based planar path following and obstacle avoidance of autonomous underwater vehicle. In *ICINCO (2)*, pages 191–198, 2011.

[130] C. Yu, X. Xiang, and J. Dai. 3d path following for under-actuated auv via nonlinear fuzzy controller. In *OCEANS 2016 - Shanghai*, pages 1–7, 2016. doi: 10.1109/OCEANSAP.2016.7485727.

[131] S. Wang, Y. Shen, Q. Sha, G. Li, J. Jiang, J. Wan, T. Yan, and B. He. Nonlinear path following of autonomous underwater vehicle considering uncertainty. In *2017 IEEE Underwater Technology (UT)*, pages 1–4, Feb 2017. doi: 10.1109/UT.2017.7890302.

[132] W. Caharija, K. Y. Pettersen, J. T. Gravdahl, and E. Borhaug. Path following of underactuated autonomous underwater vehicles in the presence of ocean currents. *2012 Ieee 51st Annual Conference on Decision and Control (Cdc)*, pages 528–535, 2012. ISSN 0191-2216. URL <GotoISI>://WOS:000327200400087.

[133] Xiao Yang, Yue Shen, Kaihong Wang, Qixin Sha, Bo He, and Tianhong Yan. Path following for an autonomous underwater vehicle using gp-los. In *OCEANS 2016-Shanghai*, pages 1–5. IEEE, 2016.

[134] Filoktimon Repoulias and Evangelos Papadopoulos. Planar trajectory planning and tracking control design for underactuated auvs. *Ocean Engineering*, 34(11):1650 – 1667, 2007. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2006.11.007. URL http://www.sciencedirect.com/science/article/pii/S0029801807000509.

[135] Xiao Liang, Yuan You, LF Su, Wei Li, and Jundong Zhang. Path following control for underactuated auv based on feedback gain backstepping. *Technical Gazette*, 22(4):829–835, 2015.

[136] Jian Gao, Weisheng Yan, Ningning Zhao, and Demin Xu. Global path following control for unmanned underwater vehicles. In *Control Conference (CCC), 2010 29th Chinese*, pages 3188–3192. IEEE, 2010.

[137] Z. Chu and D. Zhu. 3d path-following control for autonomous underwater vehicle based on adaptive backstepping sliding mode. In *Information and Automation, 2015 IEEE International Conference on*, pages 1143–1147, 2015. doi: 10.1109/ICInfA.2015.7279458.

[138] Xinqian Bian, Jiajia Zhou, Zheping Yan, and Heming Jia. Adaptive neural network control system of path following for auvs. In *2012 Proceedings of IEEE Southeastcon*, pages 1–5, March 2012. doi: 10.1109/SECon.2012.6196982.

[139] H. Kawano. Method for applying reinforcement learning to motion planning and control of under-actuated underwater vehicle in unknown non-uniform sea flow. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 996–1002, Aug 2005. doi: 10.1109/IROS.2005.1544973.

[140] H. Shen and C. Guo. Path-following control of underactuated ships using actor-critic reinforcement learning with mlp neural networks. In *2016 Sixth International Conference on Information Science and Technology (ICIST)*, pages 317–321, May 2016. doi: 10.1109/ICIST.2016.7483431.

[141] Runsheng Yu, Zhenyu Shi, Chaoxing Huang, Tenglong Li, and Qiongxiong Ma. Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle. In *Control Conference (CCC), 2017 36th Chinese*, pages 4958–4965. IEEE, 2017.

[142] Sigurd A Fjerdingen, Erik Kyrkjebø, and Aksel A Transeth. Auv pipeline following using reinforcement learning. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–8. VDE, 2010.

[143] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

[144] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[145] Chris Gaskett, David Wettergreen, Alexander Zelinsky, et al. Reinforcement learning applied to the control of an autonomous underwater vehicle. In *Proceedings of the Australian Conference on Robotics and Automation (AuCRA99)*, 1999.

[146] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, Feb 2015. ISSN 0162-8828. doi: 10.1109/TPAMI.2013.218.

[147] M. De Paula and G. G. Acosta. Trajectory tracking algorithm for autonomous vehicles using adaptive reinforcement learning. In *OCEANS 2015 - MTS/IEEE Washington*, pages 1–8, Oct 2015. doi: 10.23919/OCEANS.2015.7401861.

[148] Morten Breivik and Thor I. Fossen. Guidance laws for autonomous underwater vehicles. In Alexander V. Inzartsev, editor, *Underwater Vehicles*, chapter 4. IntechOpen, Rijeka, 2009. doi: 10.5772/6696. URL https://doi.org/10.5772/6696.

[149] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[150] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[151] Shusheng Bi, Chuanmeng Niu, Yueri Cai, Lige Zhang, and Houxiang Zhang. A waypoint-tracking controller for a bionic autonomous underwater vehicle with two pectoral fins. *Advanced Robotics*, 28(10):673–681, 2014. doi: 10.1080/01691864.2014.888373. URL https://doi.org/10.1080/01691864.2014.888373.